E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

MODELO DE SIMULACIÓN DE UNA UNIDAD DE CUIDADOS INTENSIVOS

Máster Universitario en
Ingeniería Industrial

Trabajo Fin de Máster

Imanol Echeverria Busto

Cristina Azcárate Camio

Pamplona, 03/03/2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESÚMEN

El presente proyecto describe las etapas desarrolladas para la construcción de un modelo de simulación para una Unidad de Cuidados Intensivos (UCI). El objetivo principal del trabajo es construir un modelo de simulación que represente la forma en la que los médicos de la UCI toman las decisiones relacionadas con las altas de los pacientes. Para ello, se representan todos los elementos relevantes en el funcionamiento real de la UCI como los tipos de pacientes, sus patrones de llegada, los tiempos de estancia, etc. El trabajo está dividido en tres partes principales siendo la primera una introducción al proceso de construcción de un modelo de simulación. La segunda parte se centra en el modelo de simulación basado en agentes que se va a construir, detallando cada uno de los aspectos relevantes del mismo. En esta parte se describe también la implementación del modelo de simulación en el software Anylogic. Por último, en la tercera parte del trabajo se presenta un ejemplo ilustrativo, basado en la UCI del Hospital de Navarra y se estudian distintos escenarios.

This project describes the stages developed for the construction of a simulation model for an Intensive Care Unit (ICU). The main objective of the work is to build a simulation model that represents the way in which ICU doctors make decisions related to patient discharges. To do this, all the relevant elements in the actual functioning of the ICU are represented, such as the types of patients, their arrival patterns, length of stay, etc. The work is divided into three main parts, the first being an introduction to the process of building a simulation model. The second part focuses on the simulation model based on agents to be built, detailing each of the relevant aspects of it. This part also describes the implementation of the simulation model in the Anylogic software. Finally, in the third part of the work an illustrative example is presented, based on the ICU of the Hospital of Navarra and different scenarios are studied.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICAS

1. INTRODUCCIÓN

1.1. Antecedentes

Hoy en día, las unidades de cuidados intensivos (UCI) representan únicamente entre el 5% y el 10% de las camas que hay en los hospitales, pero consumen alrededor del 30% de los recursos disponibles para cuidados de pacientes urgentes y el 8% de los costes hospitalarios.

Estos datos permiten intuir que el coste asociado a la atención hospitalaria especializada en España es bastante elevado y por tanto, la utilización eficiente de estos recursos resulta sumamente importante. Cuando se habla de la utilización eficiente se tienen en cuenta aspectos tanto económicos (ocupación de las camas, recursos necesarios, etc.) como de la calidad del servicio que se proporciona.

Al igual que en muchos otros ámbitos, la simulación es una herramienta que puede facilitar la mejora de estos servicios.

1.2. Objetivos/justificación

Cuando se habla de los objetivos del trabajo se puede identificar uno que destaca por encima del resto y es el siguiente: construir un modelo de simulación basado en agentes para que pueda ser utilizado para analizar el proceso de toma de decisiones médicas en UCIs.

A lo largo de los últimos años, investigadores del departamento de Estadística e Investigación Operativa han colaborado con médicos del Complejo Hospitalario de Navarra, para resolver problemas relacionados con la gestión sanitaria y la toma de decisiones, y han utilizado herramientas de optimización y simulación para abordar dichos problemas. Hasta ahora, el trabajo realizado se ha basado en modelos de simulación de eventos discretos, que es una de las técnicas más empleadas en el ámbito de simulación. No obstante, últimamente se está difundiendo el uso en el contexto de salud de otros tipos de simulación como la simulación basada en agentes.

El departamento, dispone de un nuevo software especializado en simulación basada en agentes. Por lo tanto un objetivo de este trabajo es la construcción de un modelo de simulación basado en agentes que represente adecuadamente el funcionamiento de una UCI y que pueda ser de utilidad para futuros estudios sobre la toma de decisiones médicas en UCIs.

1.3. Estructura del trabajo

La memoria está dividida en tres partes principales. En la primera, se realiza una breve explicación teórica acerca de la simulación en general y particularizando un poco más en la simulación basada en agentes.

A continuación, en la segunda parte, se realiza una breve explicación del funcionamiento del sistema para el cual se va a construir el modelo de simulación.

Posteriormente, se realiza una introducción al software de simulación que se va a utilizar para realizar el modelo, proporcionando una serie de nociones básicas con el fin de facilitar en la medida de lo posible la comprensión del propio modelo. Además, se explica detalladamente cuales han sido los pasos a seguir en la construcción del mismo.

Por último, de cara a realizar una comprobación del correcto funcionamiento del modelo, se procede a particularizarlo para el caso tratado en un artículo científico publicado por investigadores del departamento de Estadística e Investigación Operativa y se analizan distintos escenarios.

2. MARCO TEÓRICO

2.1. La simulación y sus fases

La simulación es una herramienta matemática que permite representar el funcionamiento de un sistema real y realizar experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar diferentes escenarios o estrategias para el funcionamiento del mismo. En general, se trata de un conjunto de métodos y técnicas matemáticas, cuyo objetivo es imitar y reproducir el comportamiento de un sistema real. En la siguiente imagen se puede observar un diagrama de funcionamiento de la simulación.

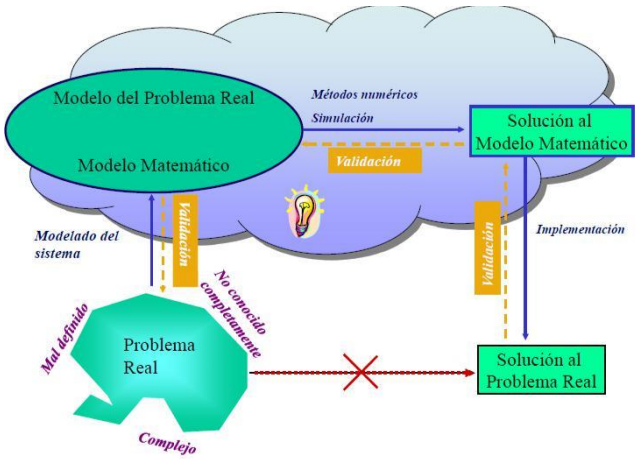


***Figura 1**: Fases de la simulación [1].*

Generalmente, se parte de un problema real que suele tener cierta complejidad y que necesita de un análisis previo para poder definirlo e intentar comprender el funcionamiento del mismo. La complejidad de estos sistemas impide poder obtener un resultado directo por lo que en muchas ocasiones son necesarias otras herramientas como por ejemplo la simulación.

El primer paso que se debe llevar a cabo es el modelado del sistema, que consiste en crear un modelo de simulación a partir del análisis de los datos iniciales disponibles. Por lo tanto, el modelado consiste en trasladar la realidad a un modelo matemático que se puede resolver mediante diferentes métodos, siendo uno de ellos la simulación. Con ello se obtiene una solución al modelo matemático basado en el sistema

real. Si los resultados obtenidos son satisfactorios se procede a la implementación del sistema real.

Una vez implementado el modelo que representa sistema real se obtiene la solución real; si dicha solución está dentro del margen de error de la solución ofrecida por el modelo matemático, se puede decir que el modelo de simulación es válido y refleja fielmente el comportamiento del sistema real.

Los ámbitos de aplicación de la simulación son muy extensos debido a que es una potente herramienta matemática que combinada con la avanzada tecnología informática de la que se dispone hoy en día permite analizar muchos tipos de sistemas reales.

Gracias a la simulación se pueden realizar predicciones realmente fiables del comportamiento de numerosos sistemas reales. En este caso, el campo de aplicación va a ser el ámbito sanitario pero también puede aplicarse en otros como:

- Diseño y evaluación de redes de telecomunicaciones.
- Aeropuertos, con el fin de ajustar los tiempos de salidas y llegadas de los vuelos. Bancos, restaurantes de comida rápida, parques temáticos, en general aquellos locales en los que se puedan llegar a generar largas colas de espera. Campo militar.
- Servicios públicos, como por ejemplo el control de los semáforos de tráfico, las planificaciones de evacuaciones de emergencia o la localización de servicios al alcance de todos.

Esta gran variedad de campos de aplicación de la simulación se debe principalmente a que existen distintos tipos de simulación. La existencia de los distintos tipos de simulación se debe a la combinación de diferentes características que la definen. Las características son las siguientes:

- Estática o Dinámica: Esta característica tiene en cuenta el tiempo en el sistema, es decir, si el comportamiento del sistema o del modelo se va a ver afectado por el paso del tiempo o no. Si el tiempo no tiene relevancia, la simulación es estática, y si por el contrario, el tiempo marca la evolución del sistema la simulación es dinámica.
- Cambio continuo o cambio discreto: Se trata de distinguir si los cambios en el comportamiento del sistema son en instantes concretos del tiempo o si por el

contrario el estado cambia continuamente. Los cambios discretos alteran el estado del sistema cuando se producen, y pueden ser aleatorios o fijos en el

- Determinística o Estocástica: Cuando es determinístico, los cambios que se producen son conocidos y constantes, es decir, se puede decir que todo es seguro y el comportamiento esta predeterminado. Por el contrario, si es estocástica, existe incertidumbre, es decir, entra en juego la aleatoriedad. Los cambios del sistema se producen en instantes de tiempo determinados por funciones estadísticas, que a su vez dependen de generadores de números aleatorios para determinar en qué instante del tiempo ocurren los eventos.

El modelo de simulación de la UCI realizado se corresponde con un modelo dinámico, de cambio discreto y estocástico.

Por otra parte, se puede decir que el proceso de simulación tiene las siguientes fases:

Planteamiento del problema

Esta primera fase y la siguiente son previas a lo que se entiende comúnmente como realizar un modelo de simulación y son tanto o más importantes como la propia fase de modelado.

En esta etapa se trata de comprender el sistema real identificando los problemas que se van a tratar de resolver. A continuación se definen las variables de entrada y salida del sistema, y los procesos y operación de los que consta el sistema real. Con todo ello se plantea un primer esquema de cómo va a ser el flujo del modelo de simulación.

Recogida de datos

En la segunda fase, se trata de recoger todos los datos necesarios para poder reflejar fielmente el sistema real en el modelo de simulación. Un buen análisis de estos datos, proporcionan gran fidelidad al modelo. En esta etapa, también se elige que datos van a ser aleatorios y cuáles serán determinísticos.

Modelado

En esta etapa se construye el modelo de simulación, dividiendo el trabajo en dos sub-etapas. La primera de ellas, trata de comprender el sistema basándose en la definición de las variables internas y en lo que sucede cada vez que ocurre un evento.

Por otra parte, en la segunda, se construye el modelo paso a paso, definiendo bien las variables, parámetros, recursos, etc.

Verificación

La verificación es un asunto de consistencia interna entre el modelo lógico y el software de programación.

Validación

Esta etapa trata de ver si hay correspondencia entre el sistema real y el modelo de simulación, es decir, si el modelo refleja fielmente lo que sucede en el sistema real. El modelo y su implementación se deben ir refinando en función de los resultados de la validación.

Simulación y Optimización

Existe la posibilidad de combinar simulación con optimización. Para llevar a cabo esta etapa en primer lugar se debe plantear el problema de optimización, definiendo el objeto de mejora y cuáles son las condiciones de la simulación. Una vez que se tiene todo bien definido y acorde con lo estipulado por el programa, se realiza la simulación repetidas veces con variables de decisión que en función de las condiciones establecidas hacen que la simulación siguiente varíe en busca de la mejor solución posible.

Análisis de resultados

En la última fase, se analizan los datos de salida de la simulación para poder comprender el comportamiento del sistema y poder dar respuestas al sistema real. En el caso de que se haya empleado la combinación de simulación y optimización las soluciones vienen dadas por el programa, ofreciendo los mejores valores en función de las restricciones y a partir de esos resultados, se debe valorar cuál de ellos se puede aplicar de la mejor manera al sistema real.

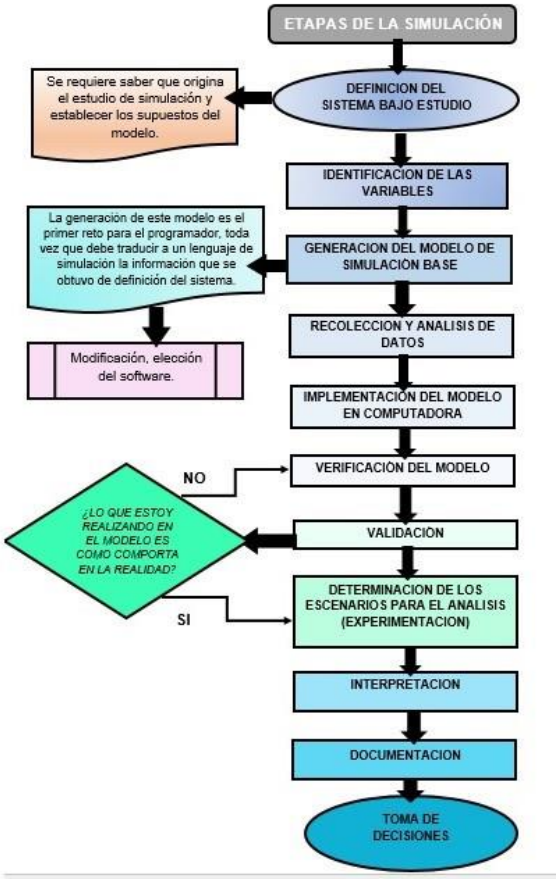


***Figura 2**: Etapas de la simulación [2].*

2.2. Modelos basados en agentes

El modelado basado en agentes es un método analítico que se aplica en las ciencias sociales desde hace relativamente poco tiempo. A pesar de ello, ha experimentado un aumento en su utilización debido a que permite construir modelos en los que se representa directamente el comportamiento de individuos así como las interacciones que mantienen en un entorno concreto de manera que el comportamiento de todo el sistema emerge como resultado de las acciones de los individuos.

Generalmente, los agentes son partes diferenciadas de un mismo programa informático que ha sido utilizado para representar el comportamiento de actores sociales (personas, individuos, organizaciones, etc.) Si se compara con otros métodos basados en variables que usan ecuaciones estructurales o diferenciales, los métodos basados en agentes permiten modelar la heterogeneidad así como la adaptación y aprendizaje de los agentes involucrados.

Hay muchas disciplinas, como la química, biología o física, donde la experimentación es el método estándar de hacer ciencia. En el caso de las ciencias sociales realizar experimentos resulta imposible debido a que no es posible asilar aquella parte del sistema donde se quiera aplicar un tratamiento específico para observar que es lo que ocurre.

2.2.1. Comparación de la modelización basada en agentes (ABM) con técnicas de modelización alternativas

Los modelos basados en agentes son especialmente recomendables en ámbitos o aplicaciones que requieran explicar un fenómeno ya que utilizar esta técnica permite identificar las causas que provocan un determinado comportamiento. Además, esta técnica es la más adecuada a la hora de representar sistemas cuyo funcionamiento es dependiente del comportamiento de los que forman dicho sistema.

Por otra parte, este tipo de modelos tienen en cuenta la localización de los agentes a la hora de su representación. A diferencia de otros tipos de modelos como los de colas o ecuaciones diferenciales, los modelos basados en agentes permiten representar la localización que tiene cada agente y utilizar dicha información para el determinar comportamiento de los agentes.

En definitiva, se puede decir que las técnicas basadas en agentes permiten identificar y analizar las causas que dan lugar a los distintos comportamientos del sistema. A pesar de ello, son técnicas complejas que requieren información detallada del sistema que se quiere analizar así como de los agentes que lo componen.

2.2.2. Elementos y características de los modelos basados en agentes

Los modelos basados en agentes están compuestos principalmente de los agentes, el entorno y las reglas de interacción entre agentes.

- Los agentes: representan los elementos activos del modelo (personas, empresas, naciones, etc.)
- El entorno: representa el espacio real en el que el grupo de agentes pueden moverse e interactuar.
- Las reglas de interacción

Por otra parte, los modelos basados en agentes tienen una serie de características que se explican a continuación:

- Heterogeneidad: cada agente puede actuar de acuerdo a sus propias reglas de interacción lo que hace que los agentes no sean similares.
- Autonomía: cada agente es independiente y es capaz de actuar solo sin ninguna forma de control centralizada por el modelo. Esto implica que cada agente toma sus propias decisiones y con sus propios criterios teniendo en cuenta la información que tiene disponible en el momento de la toma de la decisión, así como la experiencia que ha acumulado hasta el momento.
- Espacio explícito: el entorno y la localización de los agentes en dicho entorno están definidas a partir del sistema real.
- Interacciones locales: la localización que tengan los agentes condiciona las interacciones entre ellos.
- Racionalidad limitada: las acciones de los agentes están limitadas por dos aspectos. Primero por el conocimiento limitado que tienen ya que no conocen toda la información del sistema y por otra parte por el tiempo para la toma de decisiones, ya que el tiempo es acotado y no permite calcular todas las posibles soluciones a los problemas.

2.2.3. Representación de los agentes

Los agentes pueden representar cualquier tipo de componente de un sistema real empezando por elementos cuyo comportamiento puede ser representado por un número de reglas sencillas, hasta individuos con tomas de decisiones de alto nivel.

Por lo tanto, se puede decir que los agentes son submodelos. La representación de los agentes puede ser de distintos tipos como por ejemplo máquinas de estado, algoritmos de comportamiento, modelos analíticos, etc.

2.2.4. Representación del entorno

El entorno es el lugar donde conviven e interactúan los agentes del modelo y por ello es capaz de condicionar las acciones de los agentes. Además, el entorno puede ser modelado con la precisión que se desee como por ejemplo una representación multidimensional (3D) o un espacio cartesiano (2D).

2.2.5. La construcción de un modelo basado en agentes

Para la construcción de un modelo basado en agentes es necesario llevar a cabo una serie de etapas:

1- Identificación de los agentes

Es necesario identificar qué elementos del sistema van a ser modelados como agentes, puesto que cuanto mayor sea el número de agentes más complejo es el modelo.

a. Definición del comportamiento relevante: Se debe determinar cuáles son los comportamientos que son relevantes para el modelo ya que modelar el comportamiento completo de los agentes resulta complejo.
b. Modelización de agentes: Como se ha comentado anteriormente, los agentes son submodelos por lo que debe decidirse que técnica de modelado se va a utilizar para representarlos.

2- Entorno

El siguiente elemento que hay que definir es el entorno. Como ya se ha comentado anteriormente, es necesario establecer las ubicaciones donde van a estar los distintos agentes. La precisión del entorno dependerá del sistema y de qué objetivos se quieran alcanzar.

a. Identificar el entorno: es necesario identificar y establecer cuáles van a ser los límites del entorno.
b. Nivel de precisión del entorno: dependiendo del sistema que vaya a modelarse la precisión del entorno varía.

3- Interacciones

Las interacciones son las encargadas de interconectar todos los elementos del sistema, estableciendo reglas de actuación entre agentes y de agentes con el entorno.

Generalmente, las interacciones son distintas dependiendo del agente y de su ubicación. Además, se debe determinar en cada interacción que tipo de información se transfiere entre agente y agente o entre agente y entorno.

2.3. Simulación basada en agentes en el ámbito sanitario

Como se ha mencionado anteriormente, la simulación basada en agentes se ha utilizado en numerosos ámbitos como las ciencias sociales, telecomunicaciones o el ámbito sanitario.

En este último caso, se pueden observar distintos trabajos relacionados con la simulación basada en agentes y los hospitales, entre los que se pueden señalar los tres siguientes. En [5] se ha trabajado con un modelo de simulación basado en agentes para diseñar un sistema de soporte de toma de decisiones cuyo objetivo es optimizar el número de personal necesario en el departamento de urgencias minimizando el tiempo de espera del paciente y maximizando la calidad del servicio ofrecido, ~~es decir,~~ minimizando el riesgo en la salud del paciente.

En [6] se ha trabajado con un modelo basado en agentes para intentar disminuir el tiempo de espera de los pacientes en los departamentos de urgencias. Su objetivo es ver el comportamiento del sistema cuando se aplican *Fast Track Strategies* que consisten en vías rápidas de atención a pacientes dependiendo de la patología que sufran.

En [7] se trabaja con un modelo de simulación basada en agentes para determinar las políticas más óptimas para las políticas logísticas de triaje. Más concretamente, el trabajo se centra en los pacientes que sufren accidentes cerebrovasculares cuyo riesgo para la salud es elevado. El objetivo es analizar cuál es la toma de decisión tomada para los pacientes que sufren este tipo de accidentes, si se sigue la política del hospital más cercano o se traslada al paciente a la unidad de ictus.

3. DESCRIPCIÓN DEL SISTEMA REAL

En este apartado se procede a explicar de manera detallada cual es el funcionamiento del sistema real que se quiere modelar analizando los distintos elementos que componen el sistema.

Las unidades de cuidados intensivos, son áreas situadas dentro de los hospitales cuyo propósito es atender a pacientes que se encuentran en estado crítico. Generalmente, las camas y el personal especializado de estas áreas son recursos muy costosos y están sujetos a restricciones presupuestarias. Los pacientes que ingresan e estas unidades son pacientes cuyo estado de salud es de alto riesgo por lo que a la hora de diseñar estas instalaciones aparecen dos objetivos en conflicto. Desde un punto de vista económico se intenta que la ocupación de las camas de la UCI sea elevada y desde un punto de vista sanitario se intenta que siempre haya camas disponibles de manera que el porcentaje de rechazo de pacientes sea mínimo.

A continuación, se detalla cómo es una unidad de cuidados intensivos, así como los aspectos que se deben tener en cuenta para la modelización del sistema.

3.1. Descripción física

La unidad de cuidados intensivos que se quiere representar está compuesta por un gran número de elementos y espacios pero a efectos prácticos para la simulación se puede resumir en cuatro grandes zonas:

- Recepción: es la zona del complejo donde se realizan los trámites de admisión y alta de los pacientes. Los agentes que interactúan aquí son los acompañantes de los pacientes y el personal de admisión.
- Zona de camas: es la zona donde se sitúan las camas que van a acoger a los pacientes.
- Sala médica: esta sala es la sala donde están los médicos cuando no están atendiendo ninguna cama de los pacientes.
- Sala de espera: es la sala donde aguardan los acompañantes de los pacientes que han ingresado en la unidad de cuidados intensivos.

3.2. Tipos de pacientes

La unidad de cuidados intensivos del Hospital de Navarra recibe pacientes principalmente de 3 fuentes; cirugías programadas, emergencias y de planta. De estas 3 fuentes de pacientes se distinguen 8 tipos de pacientes con distintas patologías.

- Grupo 1: pacientes de cirugías programadas.
- Grupo 2: pacientes que llegan de departamentos de urgencias.
- Grupo 3: pacientes ingresados por complicaciones después de cirugía.
- Grupo 4: pacientes transferidos de las salas de enfermería.
- Grupo 5: pacientes de cirugía de emergencia.
- Grupo 6: pacientes con traumatismos múltiples que no requieren cirugía.
- Grupo 7: pacientes con traumatismos múltiples después de cirugía.
- Grupo 8: otros pacientes.

El porcentaje de pacientes da cada tipo, la tasa de llegada y su tiempo de estancia son distintos para cada uno. En la tabla a continuación se muestran estos dos datos para cada grupo de pacientes.

Grupo de paciente	Porcentaje de pacientes	Tasa diaria de llegadas (desviación estándar)	Tiempo de estancia (desviación estándar)
Grupo 1	39%	0.744 (0.75)	4.379 (8.739)
Grupo 2	14%	0.27 (0.528)	8.032 (15.276)
Grupo 3	7%	0.135 (0.372)	9.927 (14.174)
Grupo 4	12%	0.226 (0.466)	9.820 (18.694)
Grupo 5	11%	0.205 (0.449)	10.307 (12.932)
Grupo 6	10%	0.137 (0328)	10.552 (12.442)
Grupo 7	5%	0.096 (0.309)	10.677 (15.029)
Grupo 8	2%	0.054 (0.243)	5.2707 (9.5886)

***Tabla 1**: Grupos de pacientes según patología. [3]*

Teniendo en cuenta las distintas patologías que sufren los pacientes de la UCI se puede realizar una agrupación de los distintos grupos de pacientes. Por lo tanto, se agrupan estos 8 tipos de pacientes en 2 grupos principales; pacientes grupo 1 y pacientes grupo 2. Los primeros hacen referencia a los pacientes que llegan al complejo de manera no programada o aleatoria y los segundos a los pacientes que llegan de cirugías programadas y que por lo tanto su llegada se conoce con antelación.

En los próximos apartados se detalla toda la información acerca de estos dos grupos de pacientes.

3.3. Llegadas de los pacientes

En este apartado se procede a describir como es el proceso de llegada de los pacientes a la unidad de cuidados intensivos.

Pacientes grupo 1 (pacientes G1)

Como se ha comentado anteriormente, este grupo de pacientes contiene a las personas que llegan a la UCI de manera no programada, es decir, aleatoria. Estas llegadas siguen habitualmente un proceso de Poisson, cuya tasa depende de cada UCI.

Pacientes grupo 2 (pacientes G2)

Este grupo de pacientes hace referencia a los que llegan a la UCI de las cirugías programadas y por lo tanto su llegada es conocida con antelación. Las llegadas se producen de lunes a viernes, generalmente llegan a final de la mañana o a media tarde, en función del horario de las cirugías.

Para este modelo en concreto, el personal que trabaja en el hospital conoce con antelación cuántas llegadas y en qué horario se van a producir en los próximos días.

3.4. Tiempos de estancia

Como se ha mencionado anteriormente en la revisión de la literatura para los modelos de simulación de UCI, uno de los aspectos más importantes es el modelado matemático de los tiempos de estancia de cada tipo de paciente.

Para ello se han utilizado diferentes distribuciones de probabilidad: distribuciones exponenciales, Weibull, lognormal, etc. Sin embargo, este tipo de distribuciones de probabilidad no permiten representar la evolución del estado de salud de los pacientes en la UCI.

Existen otro tipo de distribuciones que permiten representar mejor esta evolución de los pacientes, las distribuciones de tipo fase.

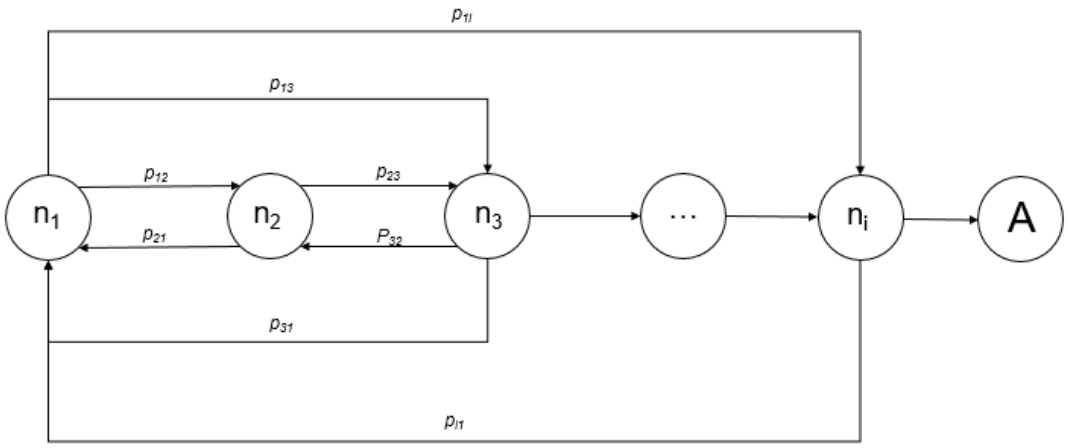
Una distribución tipo fase es la distribución del tiempo hasta la absorción en un proceso de Markov con un número finito de estados, de los cuales uno es absorbente y el resto de estados son transitorios. Las distribuciones tipo fase pueden aproximar cualquier distribución positiva. Este tipo de distribuciones han sido utilizadas en

diferentes ámbitos tales como telecomunicaciones, bioestadística, teoría de colas, etc. Este tipo de distribuciones han sido también utilizadas en el contexto sanitario, aunque generalmente sin dar un sentido médico a cada una de las fases. En [3] se propone asociar cada uno de los estados de la distribución a un estado de salud del paciente.

Por lo tanto, para modelar la estancia de los pacientes en la UCI se propone una distribución de tipo fase con n_i estados y el estado absorbente. Cada uno de los estados de la distribución representa un estado de salud del paciente. El número de fases y el tiempo esperado de los pacientes en cada uno de los estados puede variar según las características de los pacientes de la UCI que se modele. El estado absorbente representa el alta del paciente, esto es, el estado de salud del paciente que permite traspasarlo a planta sin que su salud corra ningún tipo de riesgo.

Por otra parte, las transiciones entre los distintos estados vienen determinadas por distintas probabilidades de transición. Para cada tipo de pacientes habrá que ajustar la distribución tipo fase que represente adecuadamente sus características, determinándose el número de estados, las probabilidades de transición y el tiempo medio de estancia en cada fase.

La Figura 3 muestra un ejemplo de distribución tipo fase con un número de estados n_i y unas probabilidades de transición entre estados. Estas distribuciones de tipo fase pueden tener tantos estados como se precise.



$$\begin{pmatrix} p_{11} & \cdot & \cdot & \cdot & p_{1i} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ p_{i1} & \cdot & \cdot & \cdot & p_{ii} \end{pmatrix}$$

Figura 3: *Ejemplo de distribución de tipo fase con i estados.*

3.5. Proceso de altas

En general, las altas en la UCI pueden darse de tres maneras distintas. La primera se produce cuando el paciente está estabilizado desde el punto de vista médico y por lo tanto abandona la UCI. La segunda, cuando uno de los pacientes fallece. Por último, puede ocurrir que se produzca un alta prematura de algún paciente que se encuentre en un estado de salud que permita su salida sin afectar a su mortalidad, para evitar situaciones de saturación de camas y adelantarse a las futuras llegadas de otros pacientes.

En la realidad resulta muy complejo determinar cuándo un paciente puede ser traspasado a planta y cuando no. Antes de dar de alta un paciente se deben cumplir una serie de condiciones como por ejemplo que haya una cama libre en la planta donde vaya a ser transferido, que todo el equipo médico esté de acuerdo o que la familia del paciente se comprometa en cuidar de él.

Los estados de salud de los pacientes son un concepto que engloba las diferentes etapas por las que pasan los pacientes según la severidad de su enfermedad (la Figura 3 representa la modelización matemática de estos estados). Pueden representar situaciones como por ejemplo una complicación en la salud del paciente debido a una infección o una dependencia del paciente con las máquinas sanitarias (respiración artificial, etc.).

Las altas de los pacientes pueden producirse de dos maneras. La primera de ellas es que el paciente llegue al estado de salud absorbente. Este estado representa la estabilidad del paciente y por lo tanto se considera que ha finalizado su estancia en la UCI y puede ser dado de alta sin complicaciones. La segunda manera de dar de alta un paciente es de manera prematura, es decir, sin haber finalizado el tiempo total de estancia en la UCI (sin llegar al estado absorbente). Dependiendo de las patologías de los pacientes pueden existir estados previos al absorbente que representen un nivel de salud que no conlleve demasiado riesgo para el paciente. Por ello, a la hora de considerar un alta prematura, se consideran los pacientes que estén en estos estados.

En este modelo, las decisiones de alta prematura se van a modelar matemáticamente a través de la probabilidad de dar de alta prematuramente a un paciente. Esta probabilidad depende de los elementos que influyen en la decisión de alta de un paciente en un contexto real: el nivel de ocupación de las camas, la previsión de llegadas de pacientes de cirugías programadas y el nivel de estabilidad (estado de salud) del paciente. Existen numerosas maneras de representar estas decisiones de

altas, desde una probabilidad constante hasta una probabilidad que tenga en cuenta distintas variables del modelo. Como cabe esperar, una probabilidad que se ve afectada por las variables del modelo se ajusta de mejor manera a la realidad del sistema por lo que se van a utilizar este tipo de distribuciones de probabilidad.

En [3] se propone modelar la toma de decisiones de las altas prematuras en la UCI a través de unas funciones que tienen en cuenta estos tres aspectos. Estas funciones son de tipo probabilístico y se detallan en el apartado 5.1.4.1. de la memoria. Por lo tanto, se van a programar de manera que la probabilidad de alta prematura de un paciente tenga una relación directa con el número de llegadas programadas que están previstas, la ocupación de las camas y el estado de salud en el que se encuentre el paciente.

3.5.1. Funcionamiento general

Para ilustrar el modelado de la toma de decisiones de altas, suponemos que los médicos realizan una reunión donde deciden si se va a realizar algún alta prematura. Esta reunión se produce dos veces al día, a las 8 de la mañana y a las 16 de la tarde.

Por lo tanto, el proceso de decisiones de alta se lleva a cabo dos veces al día y su funcionamiento es el siguiente. En esta reunión, los médicos comprueban el estado de salud de los pacientes y determinan los pacientes candidatos a ser dados de alta de manera prematura. Teniendo en cuenta el nivel de ocupación de camas, las llegadas de pacientes previstas desde cirugía programada y basándose en su experiencia personal deciden a qué pacientes se les da el alta de forma prematura y a cuáles no.

Como se ha comentado previamente, al definir las distribuciones de tipo fase de los pacientes se están definiendo cuales son los estados en los que un paciente puede ser considerado para un alta prematura.

Por lo tanto, para simular esta toma de decisiones de altas, seguiremos el siguiente procedimiento: en primer lugar se debe comprobar cuál es el número de pacientes (N) que hay en ese instante en los estados en los que un paciente puede ser candidato a alta prematura. Si no hay ningún paciente en ese estado, se acaba el proceso. Si hay uno o más de un paciente, se escoge uno de ellos y se realiza el sorteo con las funciones probabilísticas mencionadas anteriormente y cuya explicación viene dada en el apartado 5.1.4.1. Este sorteo se realiza entre un número aleatorio de la distribución uniforme y la probabilidad de alta prematura. Si el sorteo sale si, entonces

el paciente que se había escogido finaliza su estancia en la UCI, se actualizan las variables y se vuelve a comprobar si hay algún otro paciente en estado previo al estado absorbente. Si el sorteo sale no, se actualizan las variables correspondientes y se vuelve a comprobar si hay otros pacientes en el estado previo al estado absorbente.

Por otra parte, con el fin de que el modelo se asemeje en lo máximo posible a la forma en la que estas decisiones se toman en contexto real se va a suponer que la salida de los pacientes de la UCI no es inmediata. Por ello, se van a modelar el tiempo que se requiere para el alta del paciente (papeleo, etc.) y el tiempo requerido para la preparación de la habitación que ha dejado el paciente y que va a ser ocupada por otro paciente.

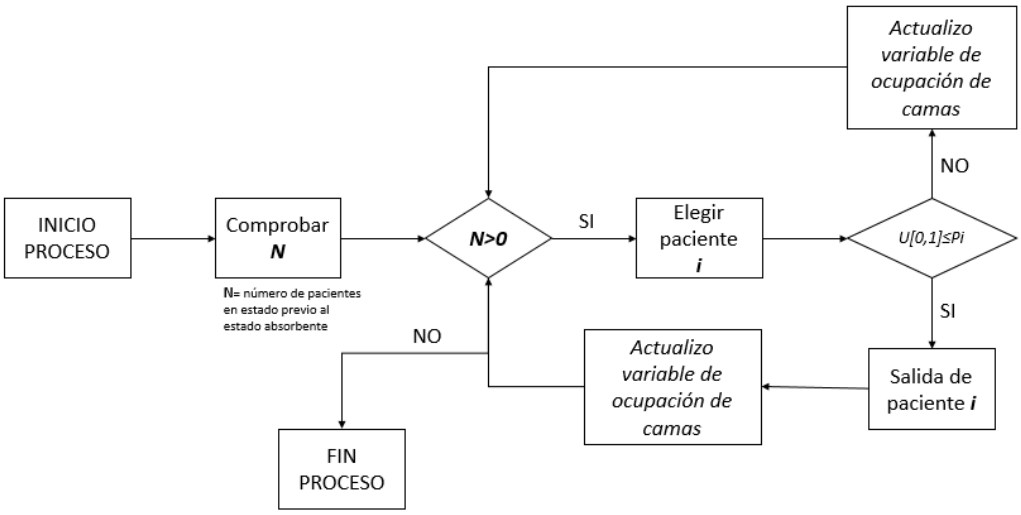


Figura 4: *Diagrama de flujo del proceso de decisión de altas prematuras*

3.5.2. Función probabilística

Para realizar el sorteo que aparece en la figura 4, este modelo va a utilizar la función probabilística que se utiliza en [3]. A continuación se detalla dicha función.

La función probabilística utilizada para modelar las altas (ver detalles en [3]) tiene en cuenta tres elementos: la ocupación de las camas *(y)* en ese momento, el estado de salud del paciente *(k)* y el número de llegadas de pacientes de cirugía programada *(X)* previstos para los próximos días. De este modo, se define la probabilidad de alta prematura como la probabilidad de alta para un paciente en estado $k \epsilon DS$ (siendo DS el conjunto de estados en los que un paciente podría ser dado de alta

prematuramente de la UCI) cuando hay *y* camas ocupadas y con una previsión para los próximos días de llegadas programadas descrita por el vector ***X***. Se propone la siguiente función logística para el modelado de la probabilidad de alta prematura.

$$Pk(y, X) = \frac{1}{1 + \exp(-\beta(1, Z)')}$$

donde ***β*** es un vector de coeficientes cuyo cálculo se detalla más adelante y ***Z*** es el vector de predictores de la función logística que depende de la ocupación de las camas (***y)*** y de la previsión de las llegadas programadas de cirugía en un horizonte de *h=3* días ***X*** = $(x_1, \ldots, x_h)$.

Siguiendo [3] se define ***β(1,Z)'*** como:

$$\beta(1, Z)' = \beta_0 + \sum_{i=1}^{3} \sum_{n=18}^{20} \beta_{in} \cdot Z_{in} =$$
$$= \beta_0 + \beta_1 X_1 1_{\{y=20\}} + \beta_2 X_1 1_{\{y=19\}} + \beta_3 X_1 1_{\{y=18\}} + \beta_4 X_2 1_{\{y=20\}}$$
$$+ \beta_5 X_2 1_{\{y=19\}} + \beta_6 X_2 1_{\{y=18\}} + \beta_7 X_3 1_{\{y=20\}} + \beta_8 X_3 1_{\{y=19\}}$$
$$+ \beta_9 X_3 1_{\{y=18\}}$$

4. DESCRIPCIÓN DEL MODELO DE SIMULACIÓN

4.1. Descripción del software de simulación utilizado

Para la realización del modelo de simulación así como de los experimentos se ha utilizado el software de simulación *Anylogic 7*. Este software proporciona una gran flexibilidad a la hora de realizar modelos de simulación ya que permite crear todo tipo de modelos como modelos basados en agentes, modelos de simulación de eventos discretos, etc.

A continuación se procede a realizar una breve explicación del funcionamiento del programa.

4.1.1. Creación de un modelo

En primer lugar se debe crear un nuevo proyecto dónde van a incluirse los distintos elementos utilizados en el modelo de simulación. La figura 7 muestra una vista principal de la interfaz donde se va a trabajar dentro del programa.

La interfaz está compuesta de los siguientes elementos:

- Árbol de proyectos (amarillo)

Aquí aparecen todos los elementos que se han incluido en el proyecto. Se asemeja a la hoja de ruta de todo lo que se ha realizado en el modelo. (Agentes, bloques, parámetros, etc.)

- Paleta (rojo):

Aquí se muestran todos los elementos incluibles en el modelo. Todos estos elementos están clasificados según distintas librerías. En este caso, las más empleadas son *Process Modeling Library*, *Agent* y *Space Markup*.

- Espacio principal (azul):

En este espacio se muestran de manera gráfica los elementos incluidos en el modelo. Es el espacio donde se deben ir añadiendo los elementos que van a constituir el modelo de simulación. Para añadir cualquier elemento de la *Paleta* al espacio principal basta con arrastrar dicho elemento a este espacio.

- Propiedades (naranja):

Es la ventana que se utiliza para modificar y ajustar todos los parámetros de los elementos del modelo. Realizando un clic en cualquier elemento situado en el espacio principal aparecen las distintas propiedades de cada elemento.

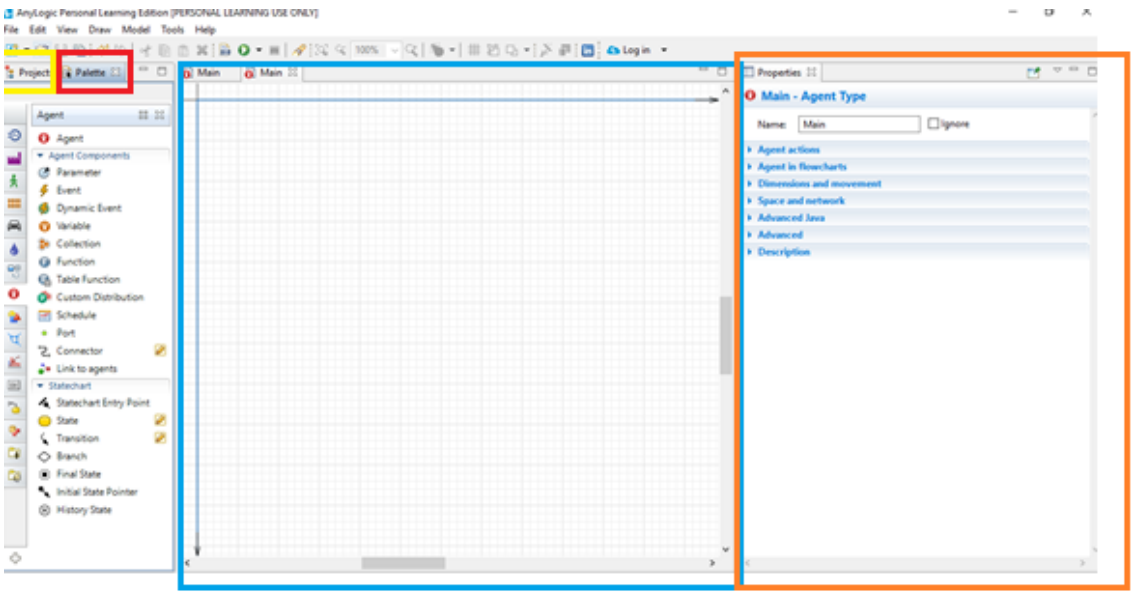


Figura 5: *Interfaz o pantalla principal del software Anylogic 7.0.*

4.1.2. Creación de agentes

Como se ha mencionado anteriormente, el primer paso para la creación del modelo es identificar cuáles son los agentes que van a participar en él. En este tipo de modelos cualquier elemento del sistema que quiera representarse debe ser un agente. En este modelo los agentes que se van a utilizar son los siguientes:

- **Pacientes**
- **Camas**

Para crear un agente es necesario arrastrar desde la *paleta* hasta el *espacio principal* el elemento agente. (Figura 6).

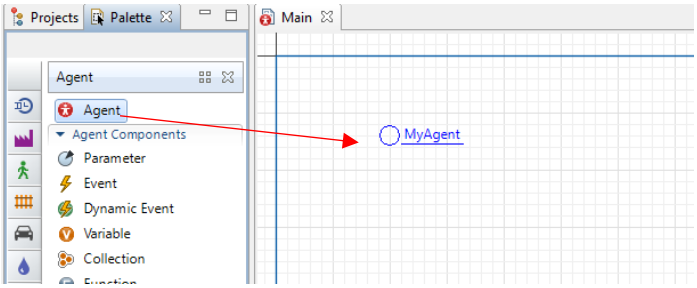


Figura 6*: Procedimiento para la creación de un agente.*

Paso 1

En la primera ventana, aparecen diferentes opciones en relación al tipo de agente que se quiere crear. Dependiendo de esto, se escoge una de las siguientes opciones.

- *Population of agents:* esta opción se escoge cuando el agente que se quiere crear forma parte de un grupo más grande de agentes, es decir, de una población como por ejemplo personas, consumidores, pacientes, etc.
- *A single agent:* esta opción se escoge cuando el agente que se quiere crear es un agente único como por ejemplo, una fábrica, un almacén, etc.

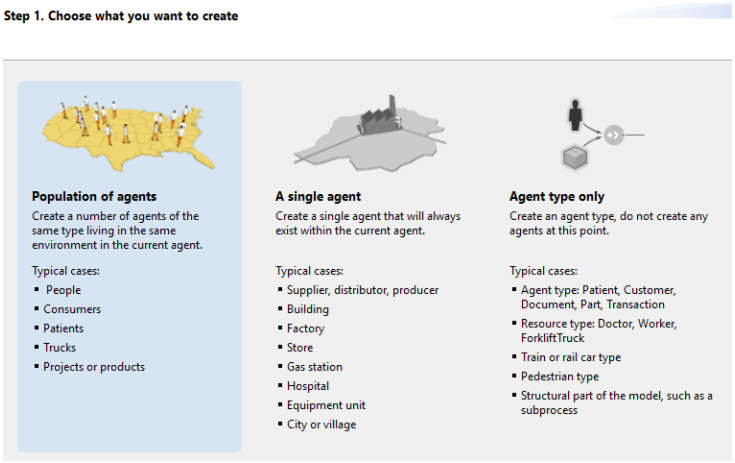


***Figura 7**: Interfaz correspondiente al paso 1.*

Paso 2

El siguiente paso es nombrar el agente que se ha creado. Además, el programa permite la opción de usar una base de datos para establecer distintos parámetros a cada agente.

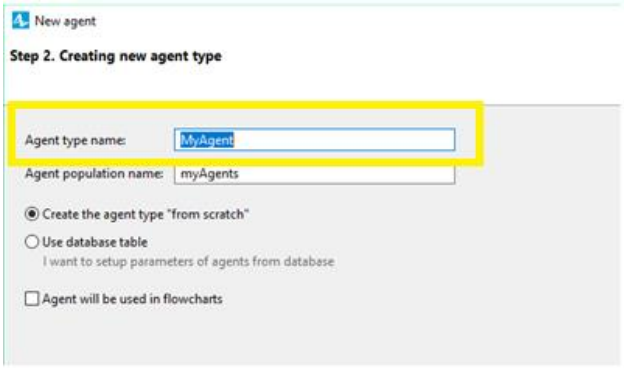


***Figura 8:** Interfaz correspondiente al paso 2.*

Paso 3

En la siguiente ventana se puede escoger cual va a ser la figura o imagen que representa cada agente concreto.

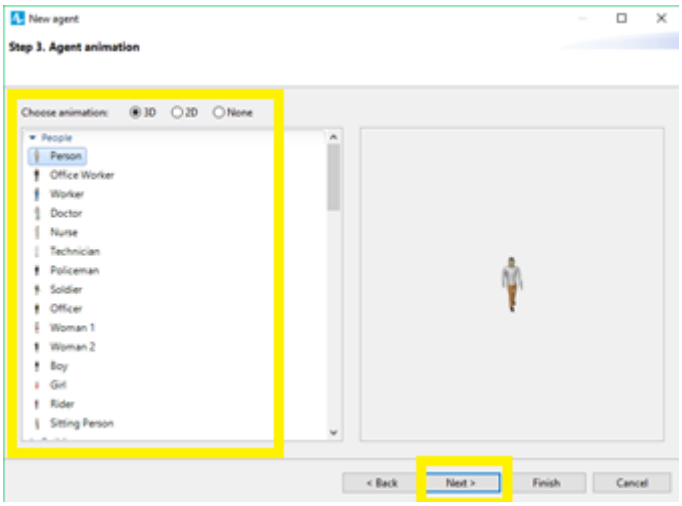


Figura 9: *Interfaz correspondiente al paso 3.*

Paso 4

Por otra parte, se puede definir cuál o cuáles van a ser los parámetros o características que van a tener los agentes.

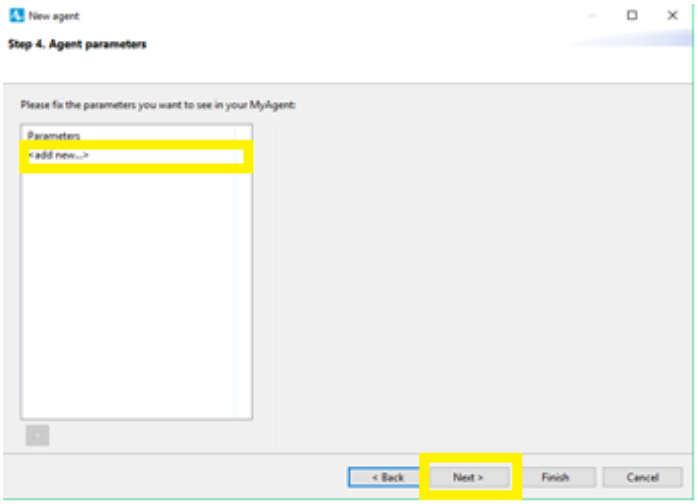


Figura 10: *Interfaz correspondiente al paso 4.*

Paso 5

Aquí, se debe introducir el tamaño que va a tener la población de agentes o si por el contrario inicialmente la población está vacía. Esto quiere decir que la población se va creando a medida que se crean los agentes.

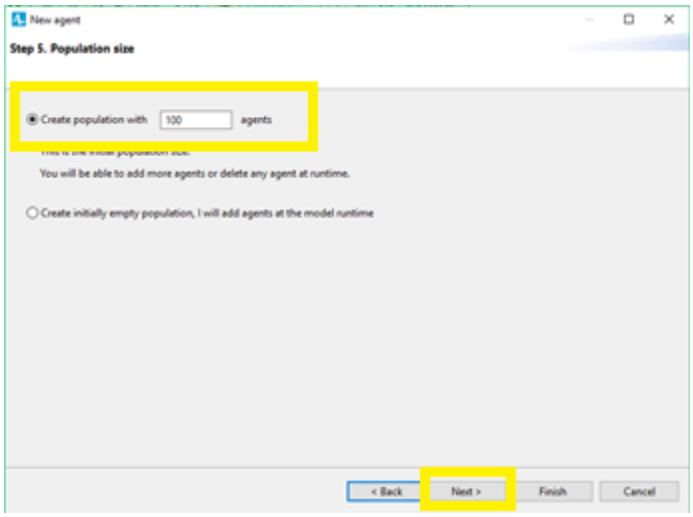


Figura 11: *Interfaz correspondiente al paso 5.*

Paso 6

Por último, se debe introducir cual va a ser el entorno en el que va a interactuar el agente. Es conveniente que este apartado sea el mismo en todos los agentes para que no haya problemas en la simulación.

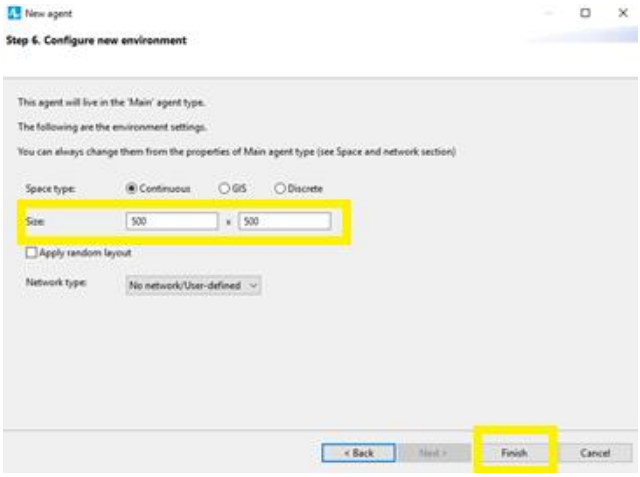


Figura 12: *Interfaz correspondiente al paso 6.*

Para poder dar por finalizada la creación de un agente se deben configurar sus parámetros. Para ello, se debe seleccionar cada agente en el espacio principal e ir modificando los parámetros en la ventana de propiedades.

Cabe destacar que en este apartado de propiedades (figura 13) se puede modificar la configuración inicial que se ha realizado cambiando aspectos como; si es un único agente o una población de agentes, el tamaño inicial de la población o el entorno donde van a moverse. Además, puede establecerse cuál va a ser la ubicación del agente en el momento de su creación.

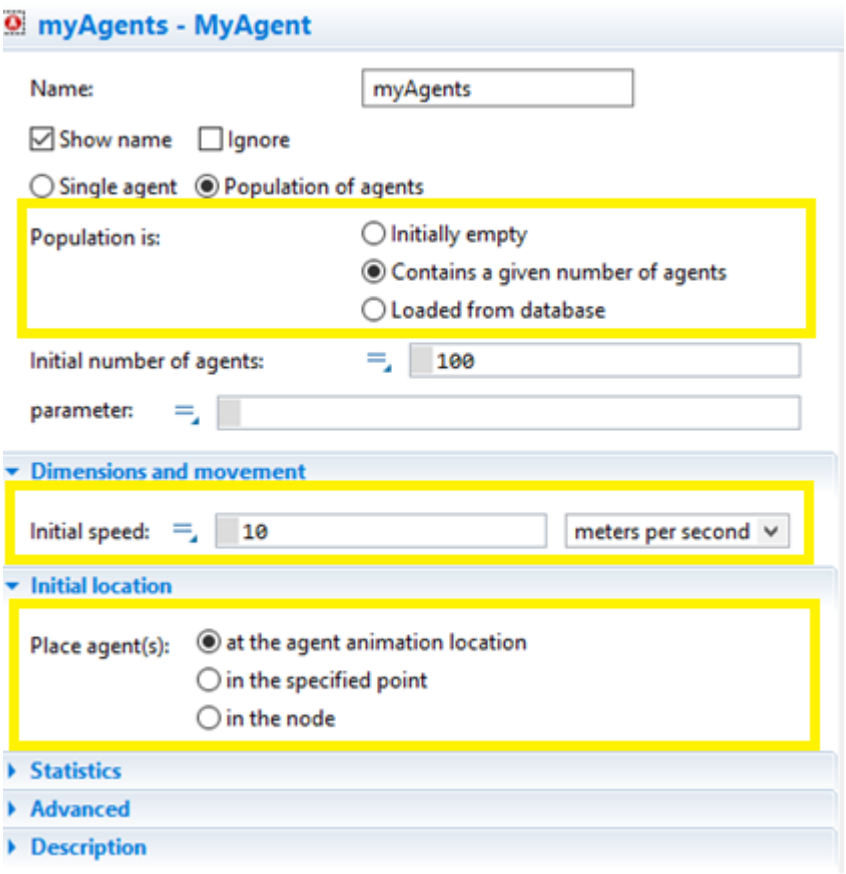


Figura 13: *Apartado correspondiente a las propiedades de un agente.*

4.1.3. Creación del flujo

Una vez creados los agentes del modelo es necesario establecer el proceso que van a describir. Para ello se va a utilizar la librería de *Process Modeling Library*. Aquí se van a encontrar todos los elementos necesarios para poder modelar el flujo de pacientes en la UCI. Dichos elementos son los siguientes:

- *Source*
- *Sink*
- *Delay*
- *Select Output*
- *Split*
- *Move To*
- *Resource Pool*
- *Seize*
- *Release*
- *Batch*
- *Time measure start*
- *Time measure end*

Al igual que ocurre con los agentes, para utilizar cualquier elemento de esta librería basta con arrastrarlo desde la *paleta* hasta el *espacio principal* del programa. A continuación se realiza una explicación de los bloques que más se van a utilizar.

Un aspecto importante de todos estos bloques es que disponen de un apartado llamado *Actions* donde puede programarse cualquier cosa, desde cambios en los valores de las variables hasta acciones concretas de los agentes. Dependiendo de los puertos de entrada y salida que tenga el bloque, esta sección tiene más o menos apartados para programar (Figura 14).

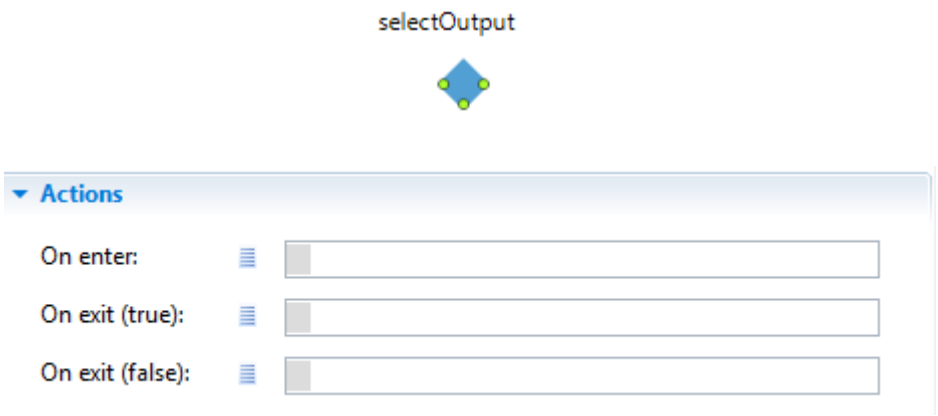


Figura 14: *Apartado Actions, común a todos los bloques del software.*

4.1.3.1. Source

Este bloque es el encargado de generar los agentes. Las opciones más importantes que se deben configurar son:

- *Arrivals defined by:* hace referencia a como se generan los agentes; con una tasa de llegadas, tiempo entre llegadas, horario específico, etc. Dependiendo de la opción que se escoja el apartado *arrival rate* varía. También, pueden definirse los parámetros de los agentes desde una base de datos.
- *Multiple agents per arrival:* aquí se debe introducir el número de agentes que llegan cada vez.
- *Location of arrival:* se debe indicar donde aparecen los agentes que se crean. Este aspecto es bastante importante de cara a realizar una animación del modelo.

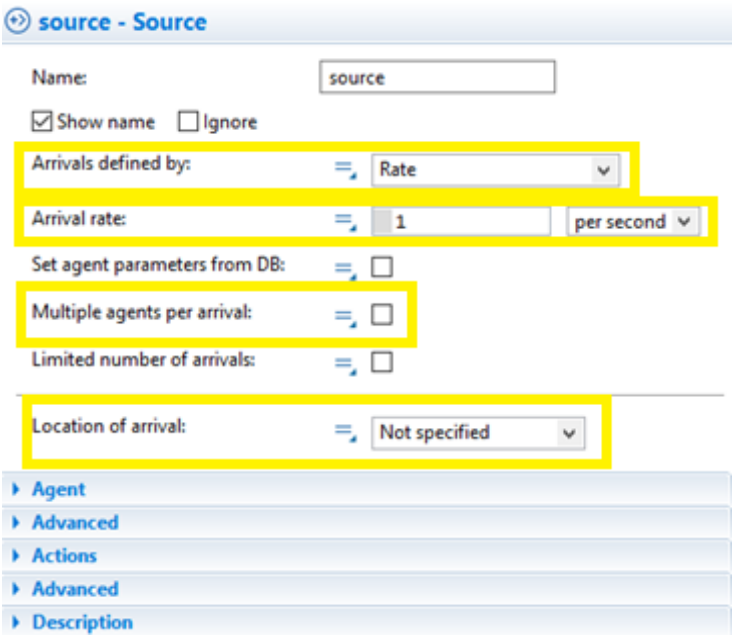


***Figura 15**: Pantalla correspondiente al bloque Source.*

4.1.3.2. Sink

Este bloque actúa como sumidero de agentes. Se utiliza cuando en las ocasiones que se quiere representar la desaparición del agente del modelo, ya sea por fallecimiento o por salida de la UCI. En este bloque en principio no es necesario configurar nada.

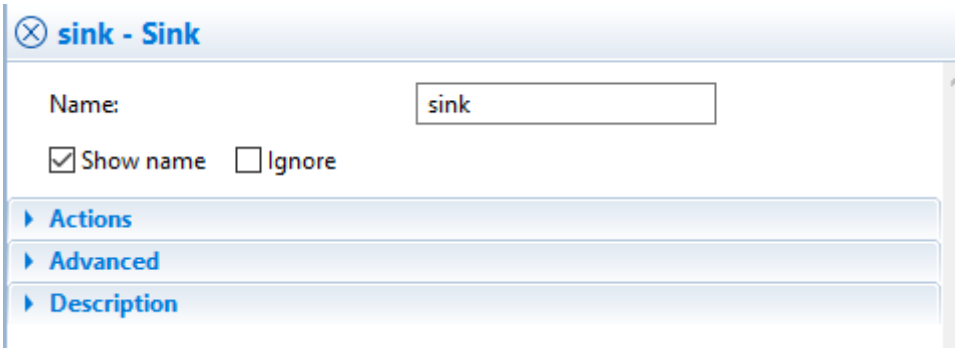


Figura 16: *Pantalla correspondiente al bloque Sink.*

4.1.3.3. Delay

La función principal de este bloque es retener un agente un determinado instante de tiempo. Dicho instante de tiempo (*Delay time)* puede ser de distintas maneras,

- Distribución de probabilidad: puede ser uniforme, triangular, exponencial, etc.
- Constante: un valor determinado en segundos, minutos, horas, etc.

Además, pueden especificarse los siguientes aspectos:

- *Capacity*: número de agentes que admite el bloque
- *Agent location:* donde se sitúa el agente mientras espera.

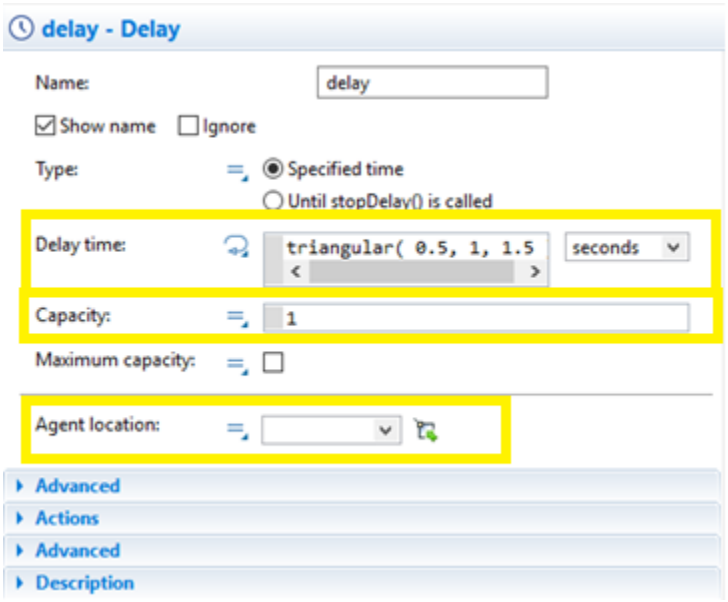


Figura 17: *Pantalla correspondiente al bloque Delay.*

4.1.3.4. Select Output

La función de este bloque es enviar los agentes por distintas salidas. El programa Anylogic proporciona dos tipos de bloques de este tipo, uno con 2 salidas y otro con 5 salidas.

Este bloque puede programarse de dos maneras. La primera es que la salida que toma el agente viene determinada por distintas probabilidades y la segunda en cambio, es que viene dada por algún tipo de condición.

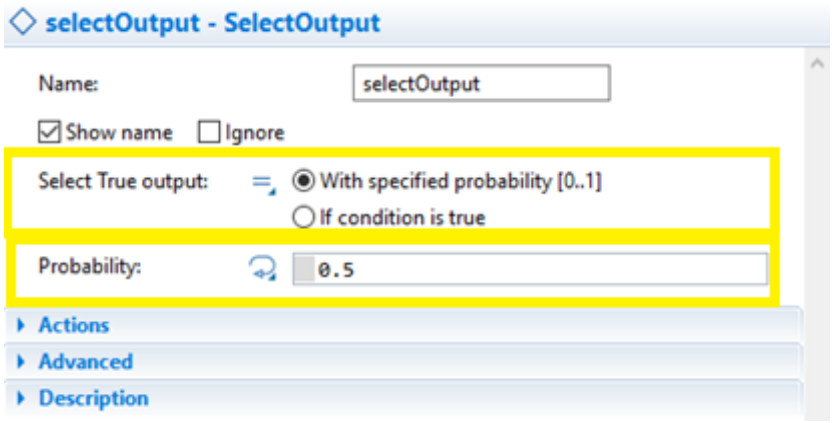


Figura 18: *Pantalla correspondiente al bloque SelectOutput.*

4.1.3.5. Split

El bloque *Split* se utiliza para crear copias de un agente. Tiene un puerto de salida por donde llega el agente en cuestión y dos puertos de salida por donde salen el original y el número de copias que se haya programado.

Para programar adecuadamente este tipo de bloques únicamente es necesario introducir el número de copias (*Number of copies*) e indicar cuál es el agente del que se quieren realizar dichas copias (*New agent (copy)*)

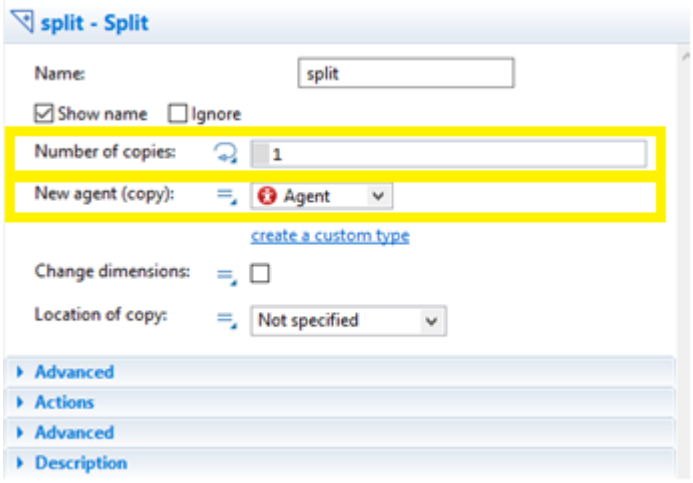


Figura 19: *Pantalla correspondiente al bloque Split.*

4.1.3.6. Move To

Este bloque es muy importante de cara a la animación del modelo. Sirve para mover los agentes de un sitio a otro, es decir, de una ubicación previamente establecida a otra como por ejemplo de la puerta de entrada a la admisión de la UCI.

La programación básica de este bloque consiste en indicar a que zona (*Node*) del modelo se quiere enviar el agente. Por otra parte, se pueden programar otros aspectos, que para este modelo no van a ser tan importantes, tales como el tipo de movimiento que lleva el agente o su velocidad.

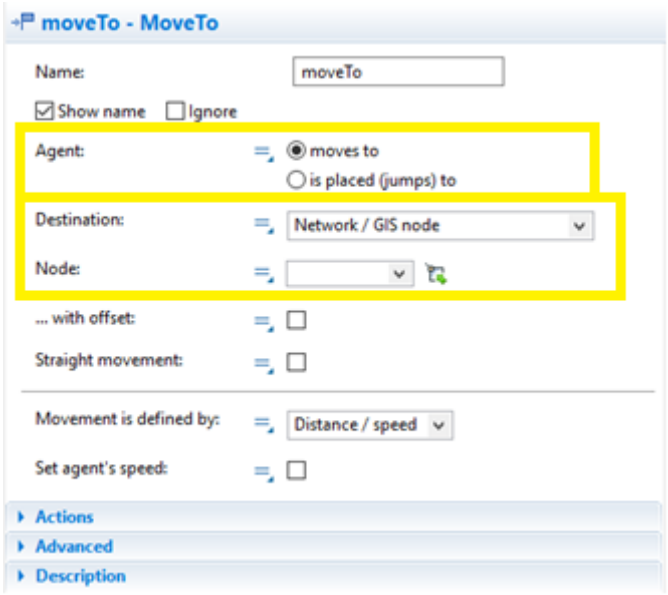


Figura 20: *Pantalla correspondiente al bloque MoveTo.*

4.1.3.7. Resource Pool

Este bloque es el encargado de asignar los recursos que van a ser utilizados en los procesos del modelo. Este bloque es muy importante ya que va a permitir establecer los agentes que van a participar en los distintos procesos. Para programarlo se deben tener en cuenta los siguientes aspectos.

- *Resource type*: se debe establecer si el recurso está en movimiento (*moving*), si es portable (*portable*) o si es estático (*static*).
- *Capacity:* la capacidad de estos recursos puede definirse de distintas maneras. En este caso se define directamente (*directly),* añadiendo directamente el número de unidades de recurso que hay.
- *When capacity decreases:* aquí se debe indicar que ocurre cuando desciende la capacidad de los recursos y se debe seleccionar la opción de que los recursos no se destruyen sino que se mantienen (*units are preservec ('End of shift')*)
- *New resource unit:* en este apartado se debe indicar que tipo de agente de los que se han creado va a ser el recurso.
- *Home location:* este aspecto es importante para la animación. Se debe seleccionar cual es la ubicación física de estos recursos.

También pueden programarse otros aspectos como el mantenimiento, los fallos o la disponibilidad de estos recursos pero para el modelo que se va a realizar no es necesario.

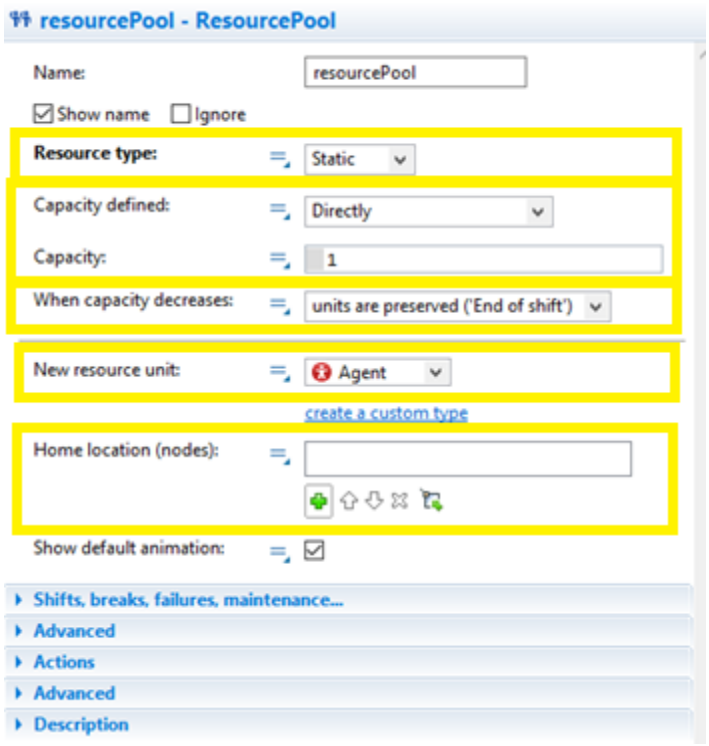


Figura 21: *Pantalla correspondiente al bloque ResourcePool.*

4.1.3.8. Seize

El siguiente bloque se encarga de juntar los recursos necesarios para cada proceso. Por lo tanto, cuando un agente entra en este bloque es retenido hasta que los recursos necesarios están disponibles y cuando lo están, se inicia el proceso.

Para programar este bloque se debe indicar cuales van a ser los recursos necesarios (*Resource sets*) para el proceso y que capacidad va a tener la cola de espera. Por otra parte, se pueden programar otros aspectos como que hacer con los recursos que se van a utilizar (*Send seized resources*) elegir el tipo de prioridad que se tiene a la hora de elegir el recurso (*Priorities*).

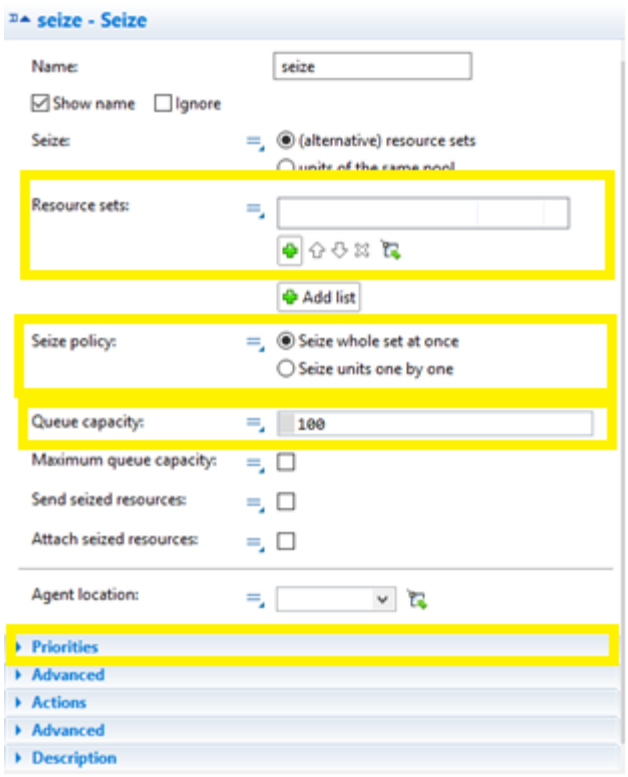


Figura 22: *Pantalla correspondiente al bloque Seize.*

4.1.3.9. Release

El bloque *release* realiza la función opuesta al bloque *seize*, es decir, se encarga de liberar los recursos que han sido utilizados por un agente en un proceso para que otro agente pueda volver a utilizarlos.

Para programarlo, se debe seleccionar la opción *All seized resources (of any pool* para que libere todos los distintos recursos que hayan sido utilizados al mismo tiempo.

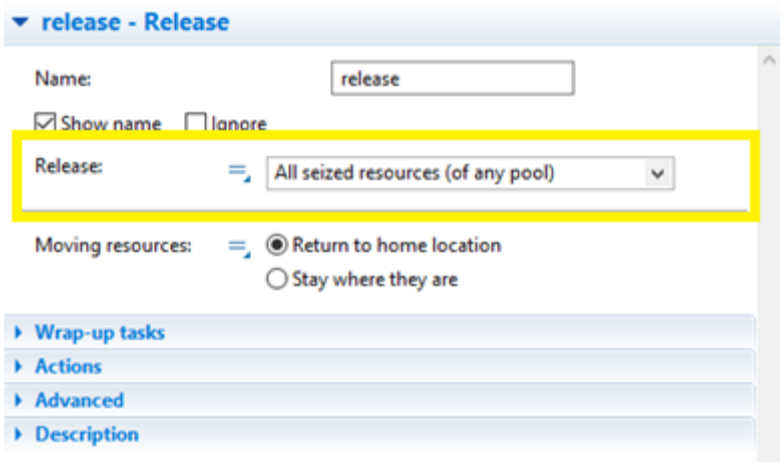


Figura 23: *Pantalla correspondiente al bloque Release.*

4.1.3.10. Batch

La función que realiza el siguiente bloque es unir un determinado número de entidades en un paquete, creando una única nueva entidad. Para programar adecuadamente este bloque es necesario indicar el tamaño de dicho paquete y cuál va a ser el nuevo tipo de agente que se va a crear.

Por otra parte, también se pueden programar aspectos como la ubicación del nuevo agente o la del propio paquete.

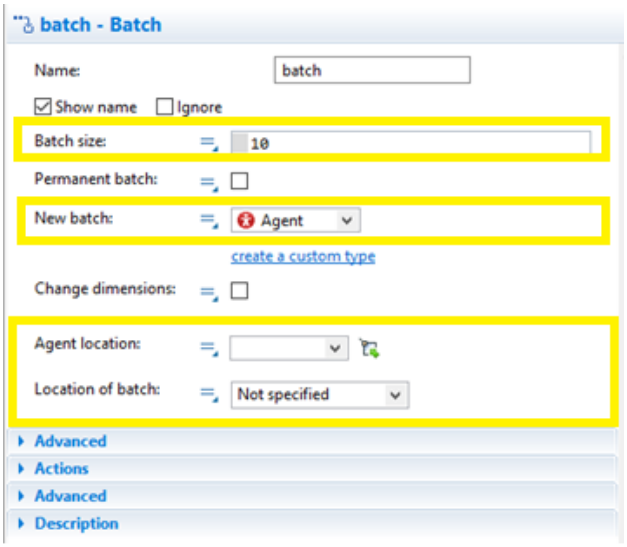


Figura 24: *Pantalla correspondiente al bloque Batch.*

4.1.3.11. Time measure start/time measure end

Estos bloques se van a utilizar principalmente para extraer resultados de tiempo de la simulación. El funcionamiento de estos bloque es sencillo ya que recogen el tiempo que pasa desde que una entidad pasa por el bloque *time measure start* hasta que llega al *time measure end*.

Para programarlos basta con situar los bloques en la zona que se quiera medir el tiempo y relacionarlos mutuamente.

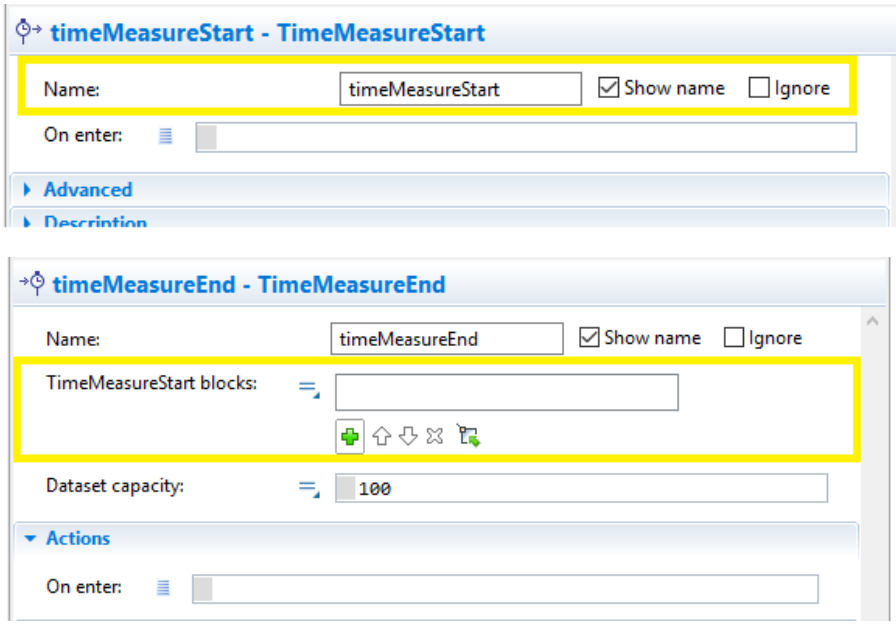


Figura 25: *Pantalla correspondiente al bloque Time Measure Start/End.*

4.1.4. Animación

Para que el modelo de simulación pueda ser comprendido más sencillamente se procede a programar una pequeña animación visual. La animación consiste básicamente en un lay-out muy sencillo de la UCI con su correspondiente zona de admisión y una zona con habitaciones conde están situadas las distintas camas.

Para programar esta animación se han utilizado los siguientes elementos que aparecen en la pestaña *Space Markup*:

- ***Rectangular wall****:* Este elemento sirve para dibujar paredes. Por lo tanto, se utiliza para delimitar cual va a ser la estructura principal del lay out que se va a representar.

- ***Path**:* Este elemento se emplea para determinar las rutas que siguen los distintos agentes del modelo. Por lo tanto, en función de la distribución en planta que se realice se dibujaran unas rutas u otras.
- ***Point node**:* Estos elementos se sitúan en las intersecciones entre las diferentes rutas *(Path)* que haya.
- ***Poligonal node**:* Con este elemento se pueden crear nodos poligonales. Los nodos poligonales son elementos que se utilizan para representar las zonas donde van a situarse los agentes del modelo. En este modelo en particular se van a utilizar para representar la zona de admisión, las camas y las salas o despachos de los médicos además de las puertas de entrada y salida de la UCI.

Por lo tanto, con los elementos que se han mencionado se procede a diseñar el lay-out de la UCI (Figura 26).

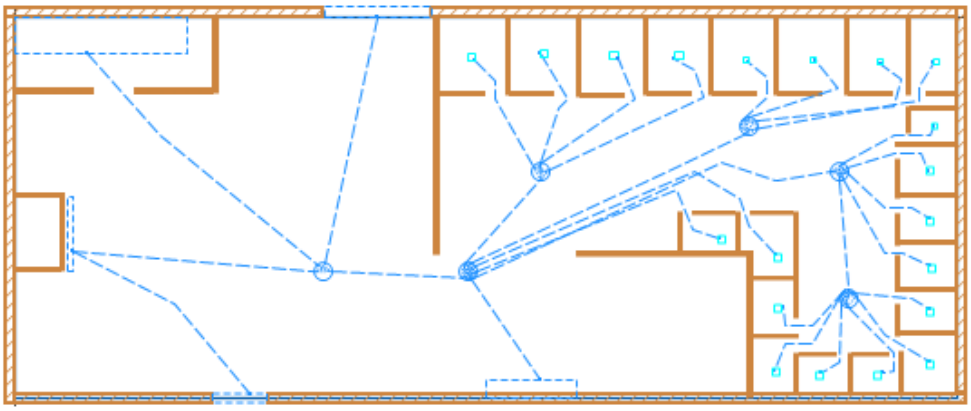


***Figura 26:** Lay-out de la planta de la UCI de la animación del modelo.*

Para empezar, se dibujan todas las paredes del lay-out empleando la herramienta *Rectangular Wall.* De este modo, se tiene delimitado el área dónde se van a mover los agentes que participan en la simulación. En la figura 27 pueden observarse todas las paredes dibujadas (señaladas mediante flechas verdes).

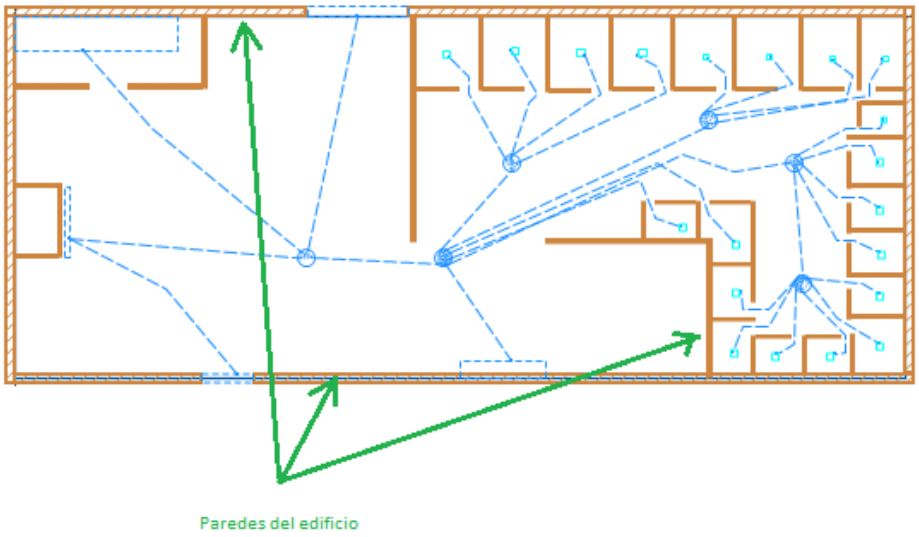


Figura 27: *Situación de las paredes (Wall) en el lay-out.*

Posteriormente, se procede a dibujar todos los puntos por los que se van a mover los agentes, es decir, los *Poligonal Node*. Empleando esta herramienta se añaden todos los elementos deseados, son los que aparecen señalados mediante flechas rojas en la imagen a continuación. (Admisión, Sala de estancia de médicos, Puerta de entrada, Puerta de salida y Camas)

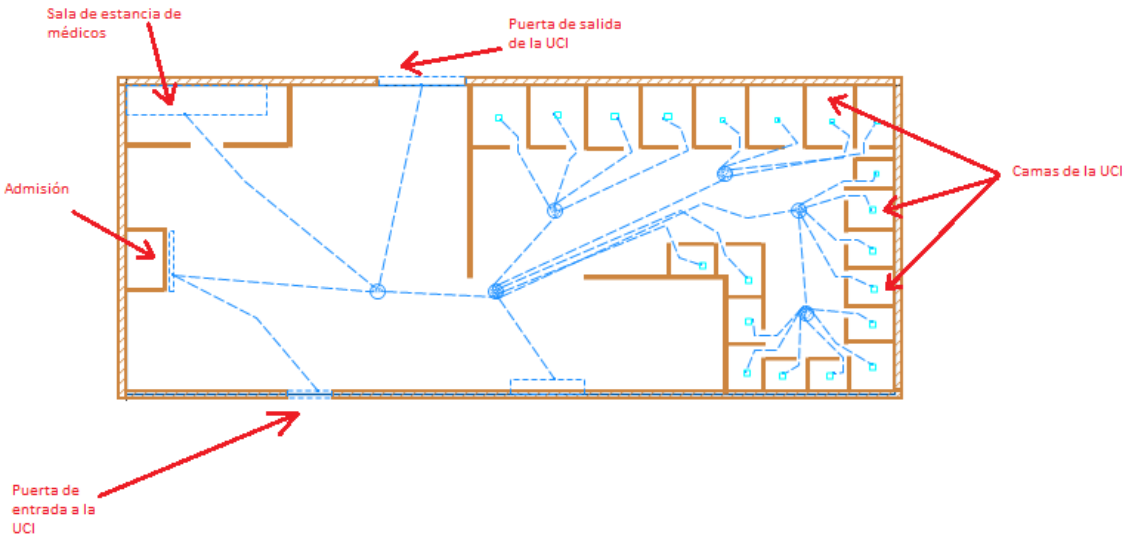


Figura 28: *Situación de los Poligonal Node en el lay-out.*

Por último, se añaden las distintas rutas *(Path)* y los distintos nodos *(Point Node)* que van a representar los caminos por donde se mueven los agentes dentro de la UCI. Para ello, basta con dibujar de un *Rectangular Node* a otro y la ruta se genera automáticamente.

Una vez dibujado el modelo, se deben configurar varios parámetros de los bloques para que participen en él.

En primer lugar, en los bloques *Source* donde se crean los pacientes, se debe indicar que los pacientes se creen en el nodo *puertaEntrada* de manera que cada vez que el bloque genere un agente en la pequeña animación que se ha creado aparezca en la puerta de entrada. Además, se selecciona una velocidad de 5 km/h para el movimiento del agente.

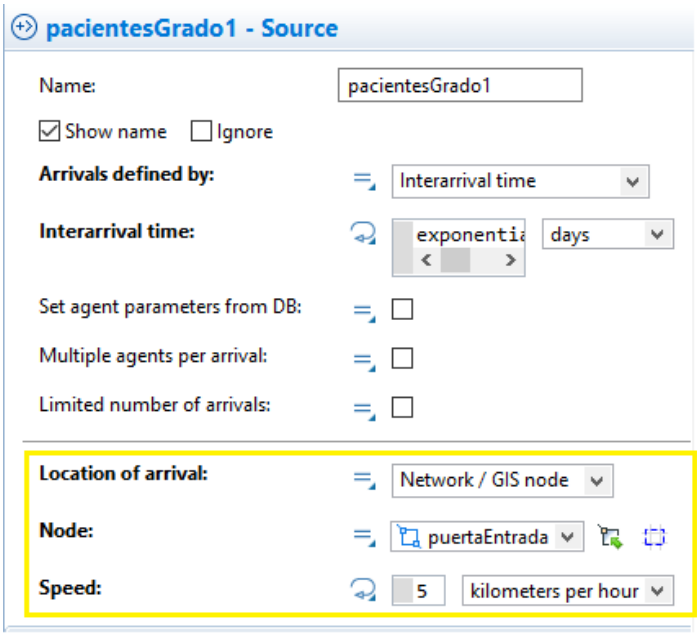


Figura 29: *Elementos a modificar en apartado Source de pacientesGrado1.*

Por otra parte, se deben modificar todos los elementos *moveTo* que hay en el modelo. Como se ha comentado anteriormente, estos bloques sirven para mover los agentes en el modelo de un lugar a otro. Por lo tanto, para que esto se vea reflejado en la animación creada se deben modificar los campos mostrados en la figura a

continuación, que hacen referencia principalmente al nodo de destino donde se va a dirigir el agente que pase por ese bloque.

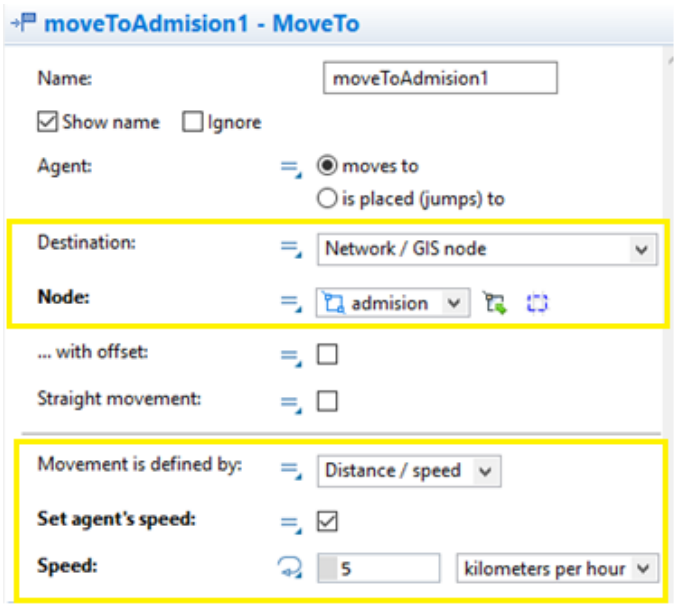


Figura 30: *Elementos a modificar en apartadso moveTo del modelo.*

Por último, se debe modificar el *ResourcePool* o unidades de recurso correspondientes a las camas. En sus propiedades, se debe indicar cuales son los nodos dónde van a situarse cada uno de estos recursos, es decir, las camas. Para ello, se deben ir añadiendo los nodos deseados en el apartado de las propiedades *Home location (nodes).*

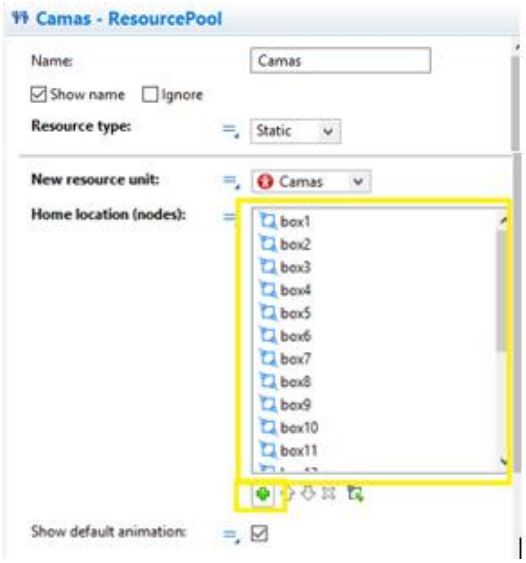


Figura 31: *Elementos a modificar en apartado ResourcePool de las Camas.*

4.2. Construcción del modelo

Una vez que se ha analizado de forma general el funcionamiento del programa se procede a implementar el modelo. La construcción del modelo se ha llevado a cabo en cuatro etapas principales; la creación de los agentes, la creación del proceso de llegadas, la creación de las estancias de los pacientes y la creación del proceso de altas. A continuación se procede a detallar cada una de estas etapas.

4.2.1. Creación de agentes

A continuación se procede a explicar cuáles son los agentes que participan en el modelo. Como se ha explicado anteriormente en la realidad existen numerosos tipos de pacientes según su patología y el número de agentes que se deben crear depende de la complejidad que se quiera dar al modelo. Por otra parte, hay que representar los recursos que van a emplear los pacientes a lo largo de su estancia en la UCI. Por lo tanto, estos son los pasos que se deben seguir para crear los distintos agentes.

- **Camas**

Siguiendo los pasos que se han explicado en el apartado 4.1.3. se crea el grupo de agentes *Cama*. Se debe indicar que el número de agentes de este tipo que hay. Por lo tanto dependiendo del tamaño de la UCI se selecciona un valor u otro.

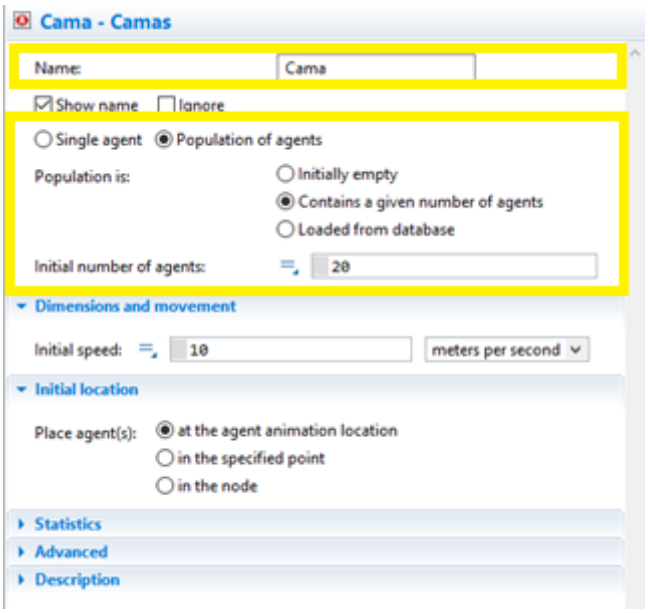


Figura 32: *Propiedades a modificar en el agente Camas.*

- **Paciente**

Este agente engloba los todos los tipos de pacientes que llegan a la UCI. En su creación, se debe indicar que la población de agentes está inicialmente vacía y que dicha población crece a medida que aumentan los agentes.

Por otra parte, se deben añadir tantos parámetros como distinciones de pacientes se deseen realizar. Para ello, al igual que se ha realizado con los agentes, se arrastra el bloque *Parameter* a la pestaña correspondiente a *paciente* como se muestra en la Figura 27.

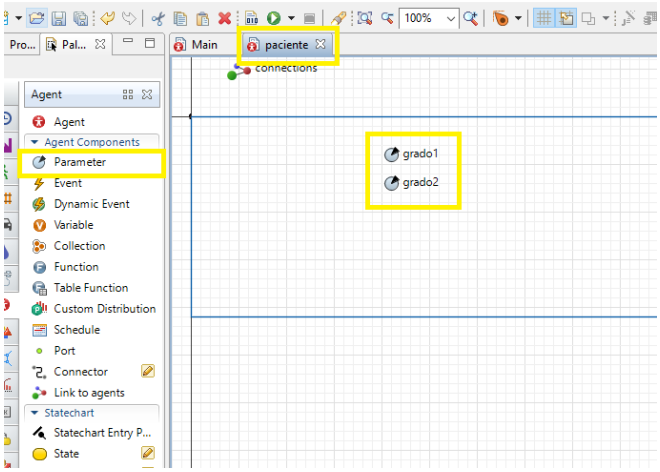


Figura 33: *Adición de parámetros al agente Paciente.*

Los parámetros deben ser boleanos debido a que van a hacer referencia al tipo de paciente que es; *gradoi==true* indica que se trata de un paciente de grado i, donde i es el número de tipos distintos de pacientes que hay.

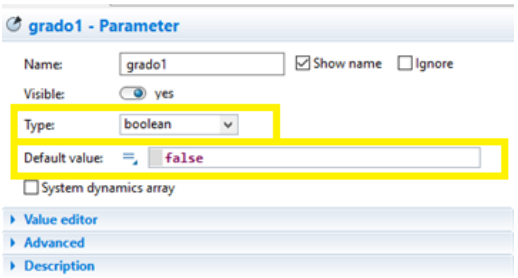


Figura 34: *Propiedades a modificar en los parámetros.*

4.2.2. Proceso de llegadas

Para que la programación del modelo sea más sencilla, se presupone que existen dos grupos principales de pacientes y cada uno de estos grupos tiene su propio proceso de llegadas.

- **Grupo de pacientes 1: llegadas no programadas (grado1)**

Se supone que este grupo de pacientes son los que llegan de manera aleatoria a la UCI por lo que su tasa de llegada puede programarse según los requerimientos necesarios (exponencial, triangular, etc.)

Para programarlo en anylogic, se utiliza el bloque Source:

- *Arrivals defined by: Interarrival time.* De este modo se programa que la tasa de llegadas viene dada por un tiempo entre llegadas.
- *Interarrival time:* Se selecciona cual es dicho tiempo entre llegadas.
- *On at exit:* Aquí se debe indicar cuál es la acción que realizan los agentes en el momento de su creación. En este caso, como se trata de los pacientes con grado de urgencia 1 se debe activar el parámetro *grado1* utilizando el comando *agent.grado1==true.*

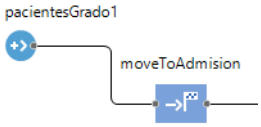


Figura 35: *Conexión de bloques para llegada de pacientes de grupo 1.*

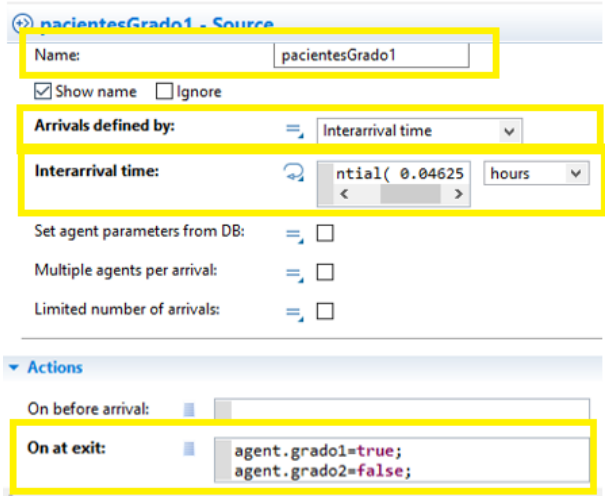


Figura 36: *Elementos a modificar en bloque Source de pacientes grupo 1.*

- **Grupo de pacientes 1: llegadas programadas (grado2)**

Los pacientes con grado de urgencia 2 (grupo 2) son los pacientes cuya llegada es conocida como por ejemplo los pacientes que llegan de cirugías programadas.

Para poder programar adecuadamente estas llegadas es necesario crear un lazo o bucle que realice el cálculo del número de pacientes que van a llegar cada día. Este bucle va a ser el encargado de calcular (mediante un sorteo de probabilidades que se explica posteriormente) cuantos pacientes de este tipo llegan cada día a la UCI.

La creación de este bucle es necesario debido a que las llegadas programadas son conocidas por los médicos de la UCI con un horizonte de 3 días. Por lo tanto se necesita crear una serie de variables que almacenen el número de pacientes que llegan a la UCI a lo largo de la semana. Cabe destacar que este tipo de pacientes llegan únicamente de lunes a viernes.

La idea general de lo que realiza este bucle básicamente es que al inicio de cada día calcula cuantos pacientes programados llegan (mediante un sorteo) y a almacena en una variable, de manera que la información de las llegadas que tenga esta variable pueda leerse desde el lazo principal del modelo, que es el que genera los pacientes programados cada día.

Las variables que participan en este bucle son las siguientes:

- **dia_sem**: variable que indica que día de la semana es.
- **ind_s**: variable empleada para actualizar el vector de llegadas programadas.
- **pred**: vector de 7 variables que contiene las llegadas programadas: pred[6] = (pred[0], … , pred[6])
- **num_sorteo_llegadas**: número aleatorio según la distribución que se desee y que se emplea para decidir el número de llegadas al día.

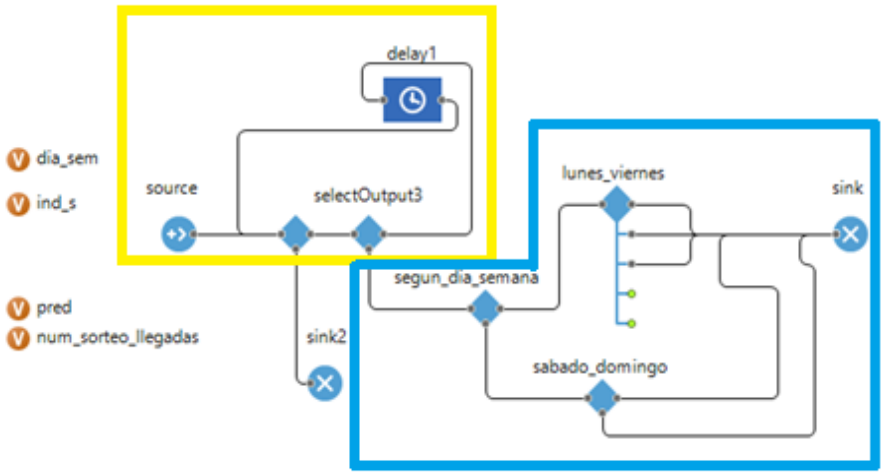


Figura 37: *Bucle correspondiente a la predicción de llegadas de pacientes grupo 2.*

Este bucle está compuesto de dos partes principales (recuadros amarillo y azul) por lo que para entender mejor su funcionamiento se procede a explicarlos por separado.

En la primera parte del bucle, la que aparece en el recuadro amarillo en la figura 37 se inicia con un bloque s*ource* que genera una entidad al día. Se debe programar el bloque para que a la llegada de cada entidad la variable *dia_sem* se actualice como si fuese un contador de los días de la semana. Por otra parte se debe poner la variable *contador_hoy* a cero.

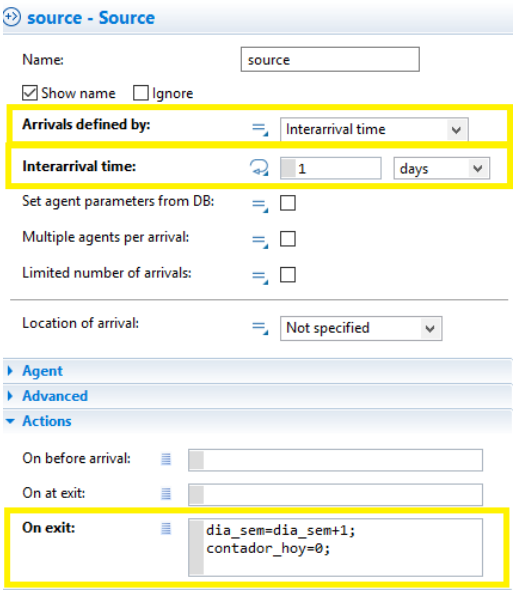


Figura 38: *Elementos a modificar en bloque Source de pacientes grupo 2.*

Las entidades generadas permanecen en el bucle superior (amarillo) del lazo debido a que se debe recorrer y actualizar el vector que contiene las llegadas para cada día y actualizar el último valor del vector. Este vector está compuesto por 7 variables, una por cada día de la semana.

Para ello, en el bloque *selectOutput3* se programa la condición de *ind_s≤5* (el vector de llegadas comienza en 0) para poder pasar al bucle inferior (azul). Así, se recorre la variable hasta que se llega al séptimo día, cuyo valor de llegadas es el que se tiene que sortear. Para actualizar el vector de llegadas se debe realizar la siguiente programación; *On exit (true)*, es decir, a la salida de la entidad por el puerto true se programa *pred[ind_s]=pred[ind_s+1]* y en el bloque *delay* que tiene a continuación se programa *ind_s=ind_s+1*. De este modo, se recorre y actualiza todo el vector de llegadas.

Una vez que la variable *ind_s* toma el valor 6 la entidad pasa al bucle inferior. Esto quiere decir que se ha recorrido todo el vector y que se deben generar las llegadas para el séptimo día, que en este caso se corresponde con la posición 6 del vector de llegadas. Cuando ocurre esto, en el apartado *On exit (false),* es decir, en el puerto false del bloque, se programa que la variable *ind_s* se pone a cero y se genera un número aleatorio.

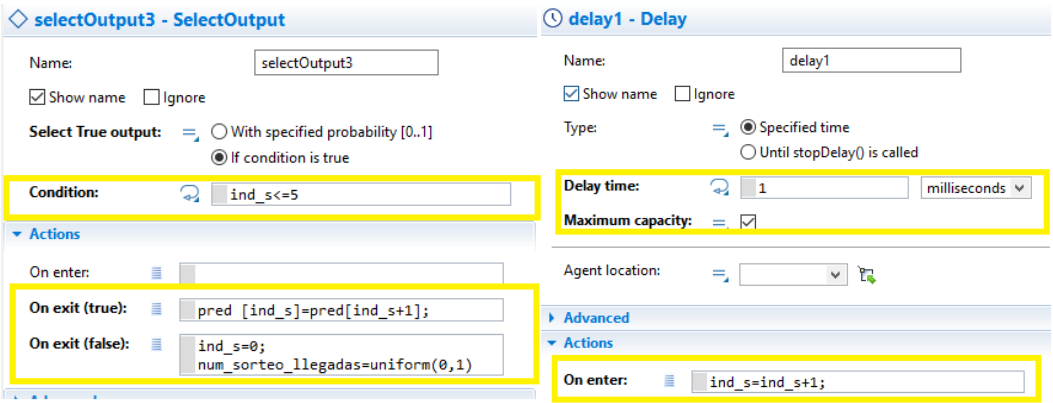


Figura 39: *Elementos a modificar en bloques SelectOutput y Delay del bucle superior (amarillo) de predicción de llegadas.*

En ese momento, la entidad pasa al bucle inferior y dependiendo del día de la semana que sea la entidad es enviada por una por una de los puertos de salida de estos bloques.

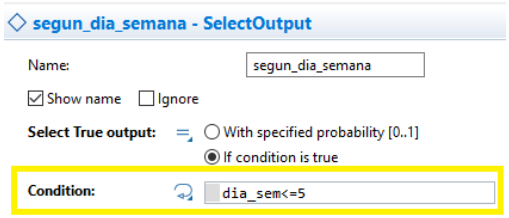


Figura 40: *Elementos a modificar en bloque SelectOutput del bucle inferior (azul) de predicción de llegadas.*

Por lo tanto, dependiendo de qué día de la semana sea el número de pacientes que llegan a la UCI varía. El número de llegadas depende de la distribución de probabilidad que se aplique al modelo por lo que puede variar según la distribución escogida.

De este modo, se procede a programar el en bucle principal las llegadas de los pacientes de grupo 2. En el bloque *source* se programan las llegadas para una llegada al día. Se debe determinar cuál es el instante de tiempo en el que llega esta entidad puesto al bucle de generación de llegadas tiene que darle tiempo a actualizar las llegadas para ese día. En este modelo, dicha llegada se produce aproximadamente a las 00:30 horas. El número de agentes que llega cada día viene dado por *pred[0],* que es el primer componente del vector de llegadas programadas mencionado anteriormente.

Por otra parte, al igual que con las llegadas no programadas, a cada agente que nace se le asignan los atributos correspondientes para identificar el agente como agente de grupo 2 (agent.grado2=true).

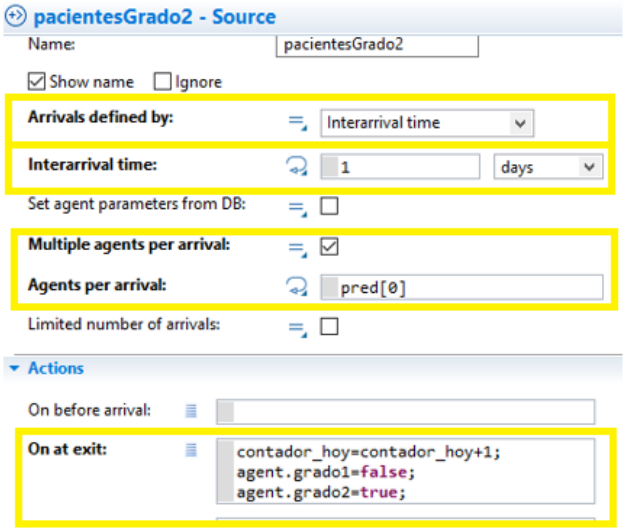


Figura 41: *Elementos a modificar en bloque Source de pacientes grupo 2.*

Con el fin de simplificar el modelo, se decide que cada paciente tenga una hora exacta de entrada en la UCI (según el número de pacientes que lleguen). Por lo tanto, se debe incorporar un pequeño lazo que simule el tiempo de llegada de los pacientes a lo largo del día.

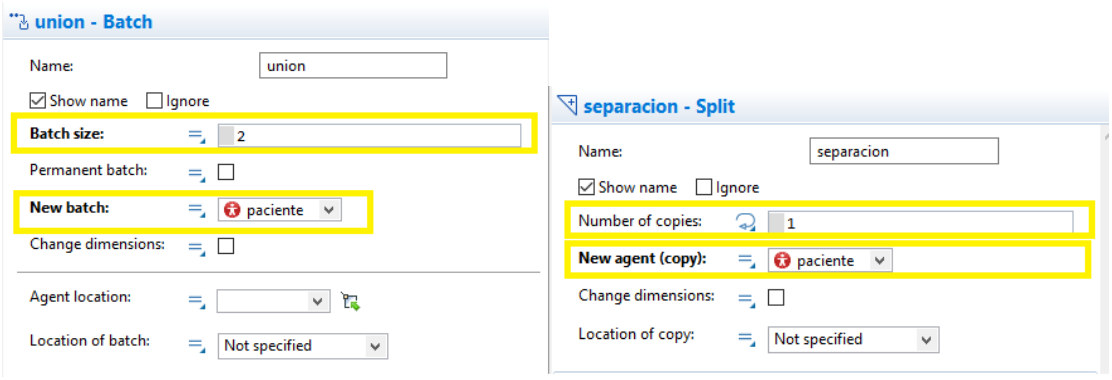


Figura 42: *Elementos a modificar en bloques Batch y Split del bucle de llegadas de pacientes grupo 2.*

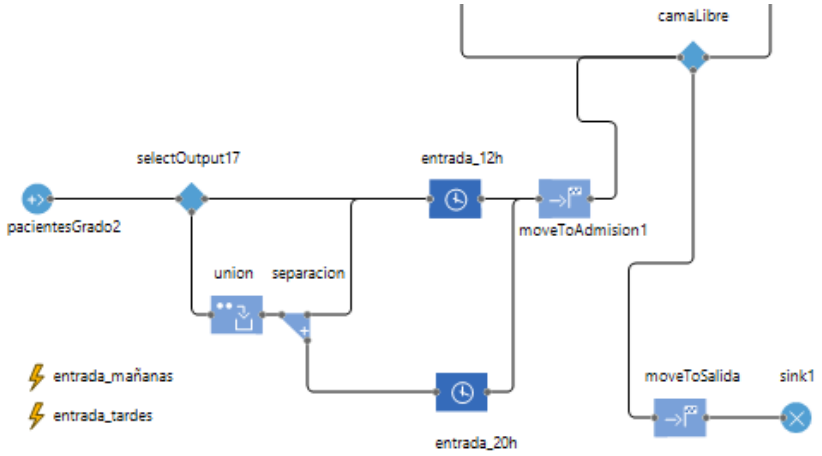


Figura 43: *Bucle de llegadas de pacientes grupo 2.*

Estos bloques, son bloques del tipo *delay* y están programados para que retenga los agentes que hay hasta que llegue el instante de tiempo en el que entren en la UCI. Este instante de tiempo es programable modificable dependiendo de la hora de entrada que se desee establecer. Para dar la orden, se crean dos eventos cíclicos que se repiten una vez al día, y que se ejecutan a la hora que se programen liberando los agentes que hay en su correspondiente bloque y permitiendo su ingreso en la UCI.

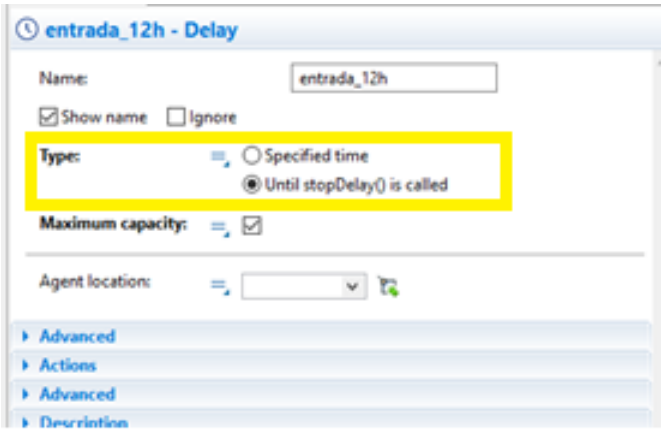


Figura 44: *Elementos a modificar en bloque Delay para la entrada de pacientes.*

Se realiza un pequeño bucle en el apartado *Action* que se ejecuta cada vez que ocurre el evento y su finalidad es liberar los agentes que contiene. El bucle comprueba si hay algún agente en el bloque *delay entrada_12* y si hay alguno, ejecuta la orden de *stopDelayForAll()* que se encarga de dar por finalizada la estancia en dicho bloque.

```
if (entrada_12h.size()>0) {

	entrada_12h.stopDelayForAll();
}
```

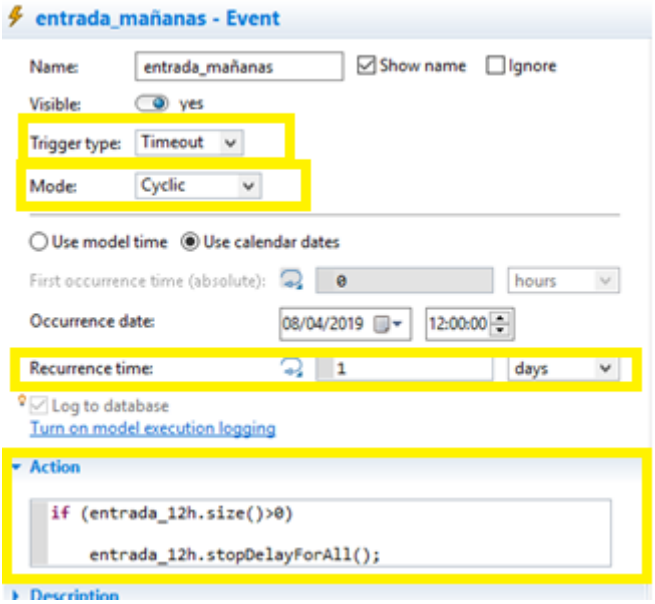


Figura 45: *Elementos a modificar en evento para la entrada de pacientes.*

4.2.3. Estancia de los pacientes

Como se ha comentado anteriormente la estancia de los pacientes se va a representar mediante distribuciones tipo fase con i estados de salud diferentes incluyendo el estado absorbente.

Para modelar cada uno de estos estados se utilizan bloques *delay* y se debe programar únicamente el tiempo de espera que corresponde a los estados del 1 al n asignándoles el tiempo que se crea conveniente.

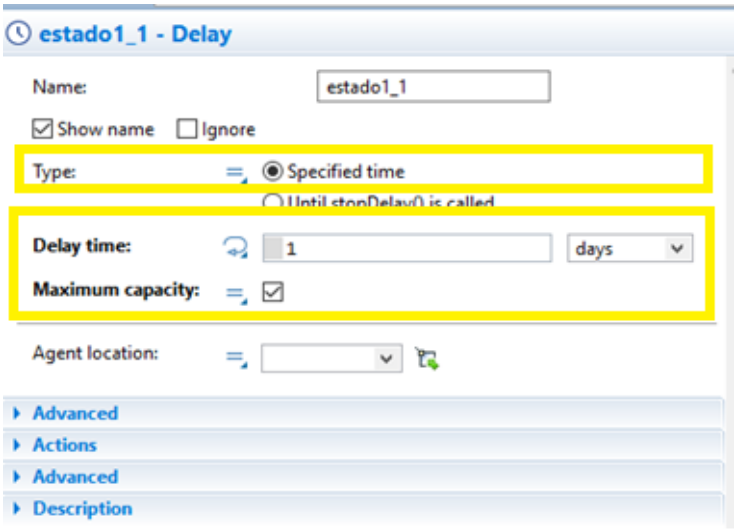


Figura 46: *Modificación del bloque Delay para tiempo de estancia en fase 1.*

El modelo está compuesto por dos tipos de pacientes y estos a su vez se subdividen en *exitus* y *non-exitus*, que hacen referencia a los pacientes que fallecen y no fallecen. Por lo tanto, el lazo principal de simulación contiene 4 lazos por donde va cada paciente.

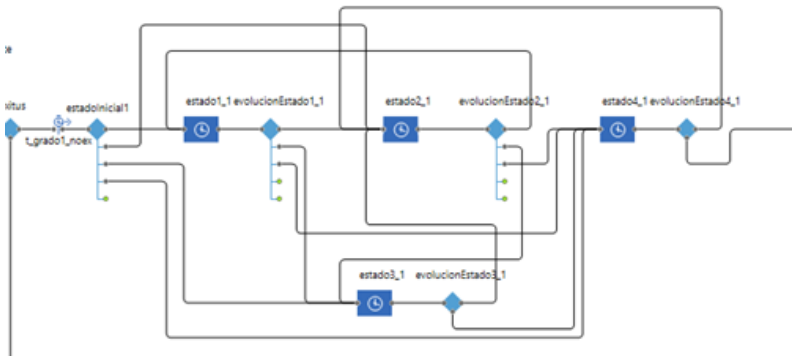


Figura 47: *Representación de la distribución de tipo fase de 5 estados mediante bloques.*

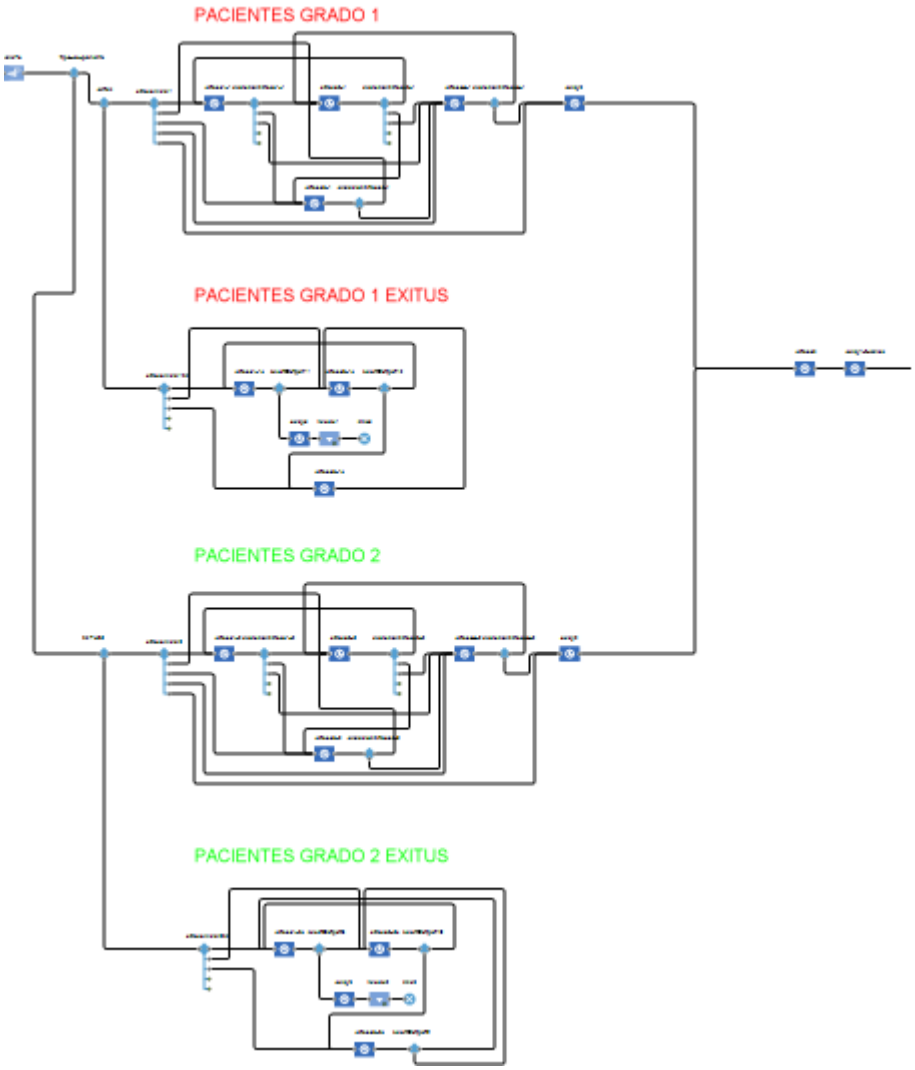


Figura 48: *Representación de la distribución de tipo fase de 5 estados mediante bloques para cada tipo de paciente.*

Para modelar las transiciones entre los distintos estados de salud de los pacientes se utilizan los bloques *SelectOutput* que permiten enviar los agentes por distintas salidas en función de distintas probabilidades. Las probabilidades de transición de un estado a otro, al igual que el porcentaje de pacientes de cada tipo que son *exitus,* son las mencionadas anteriormente en el apartado 3.5.

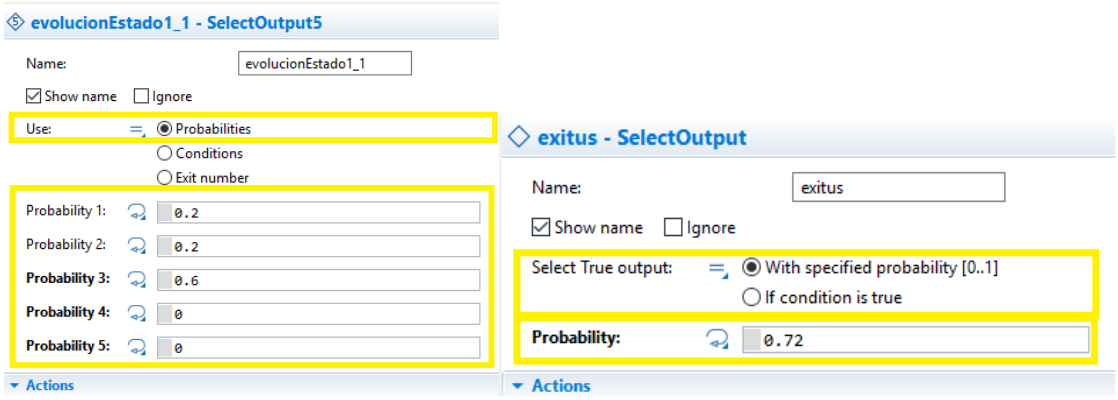


Figura 49: *Elementos a modificar para introducir las probabilidades de transición entre estados y las probabilidades de paciente exitus.*

4.2.4. Proceso de altas

Cuando se habla de las altas se pueden distinguir las altas prematuras o las que se producen sin terminar el tiempo de estancia de los pacientes y las altas normales, que son las que ocurren una vez finalizado el tiempo de estancia de los pacientes.

Para modelar el proceso que la toma de decisiones de las altas se ha realizado el siguiente bucle en el modelo. Cabe destacar que las decisiones de altas se toman dos veces al día. Los médicos se reúnen al inicio de la mañana y al inicio de la tarde para discutir y decidir las altas que se producen en la UCI.

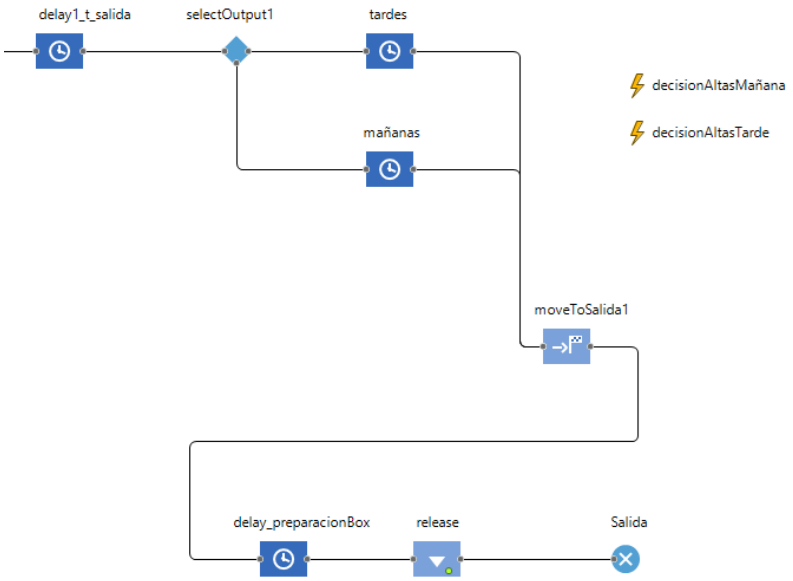


Figura 50: *Bucle correspondiente a la salida de los pacientes de la UCI.*

Una vez que los pacientes salen del último estado de salud, se ha colocado un bloque *delay* que hace referencia al tiempo de tramitación del alta del paciente. Como ya se ha explicado en apartados anteriores, realizar el alta de un paciente no es algo instantáneo y requiere un tiempo. En este caso se ha programado el bloque *delay_t_salida* con un tiempo *setupTime1* en horas. Se ha creado un parámetro (*setupTime1*) para modelar este tiempo.

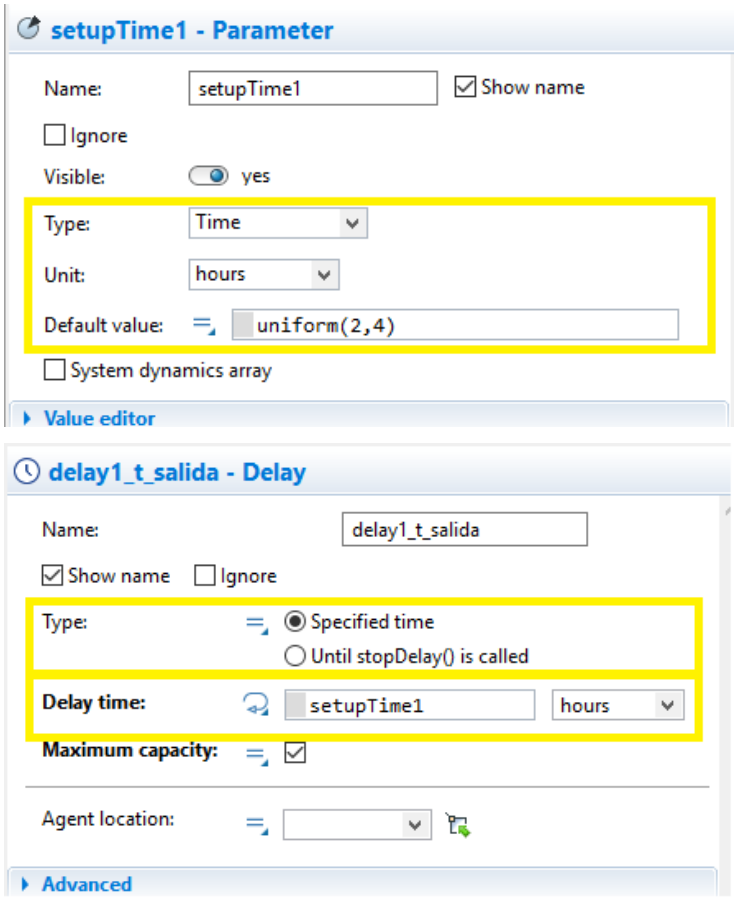


Figura 51: *Elementos a modificar en bloques Parameter y Delay para modelar el tiempo de tramitación de alta de pacientes.*

Como se ha comentado anteriormente, en este modelo se ha supuesto que las altas se tramitan por la mañana y por la tarde. Por lo tanto, cuando ha finalizado este tiempo de tramitación de altas dependiendo de la hora que sea el paciente ira al bloque *tardes* o al bloque *mañanas.*

Para establecer la hora exacta en la que se tramitan las altas, al igual que se ha hecho con la entrada de los pacientes a la UCI, se vuelven a programar dos eventos cíclicos que se repiten una vez al día (*decisionAltasMañana y decisionAltasTarde*) que ocurren a las hora que se desee para que liberen los agentes que hay en los bloques de salida de la UCI. Esto lo realizan gracias al siguiente bucle de programación:

```
if (mañanas.size()>0) {

	mañanas.stopDelayForAll();
}
```

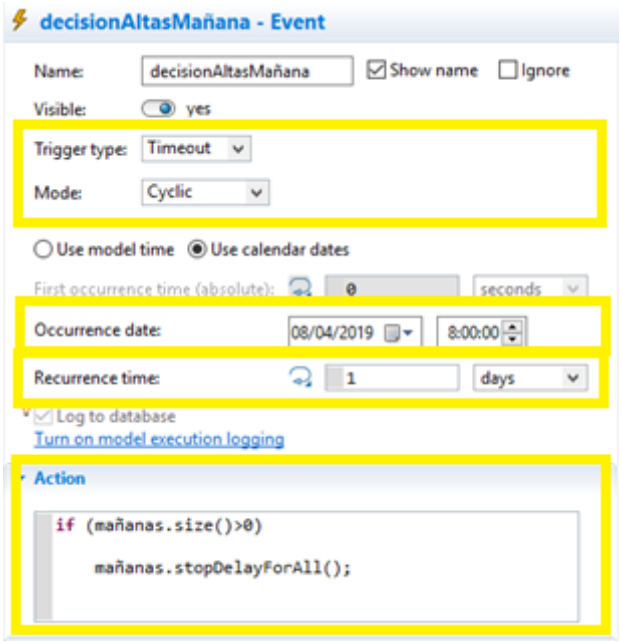


Figura 52: *Elementos a modificar en el evento que ejecuta las altas.*

A continuación, cuando los agentes han sido liberados de estos bloques se supone que han sido dados de alta y pasan a otro bloque *delay* (*delay_preparacionBox*) que se corresponde con el tiempo de preparación de la habitación y la cama que han quedado libres. Este tiempo viene recogido en un parámetro llamado *setupTime2* y puede modificarse dependiendo de las necesidades del modelo.

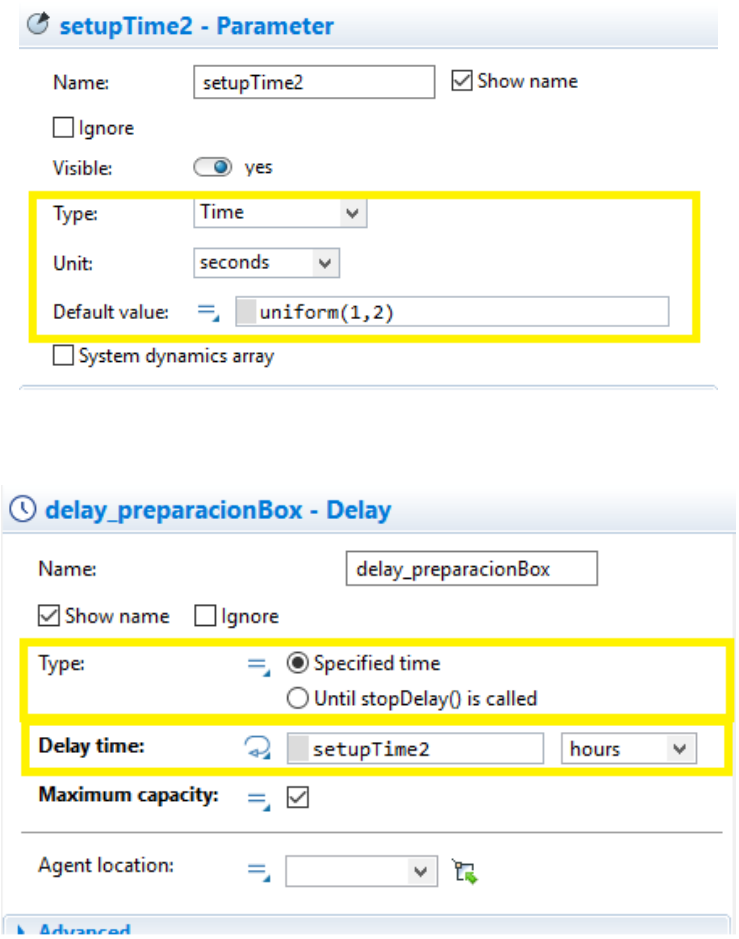


Figura 53: *Elementos a modificar en bloques Parameter y Delay para modelar el tiempo de preparación de camas.*

Por último, los agentes deben pasar por un bloque *release* que se encarga de liberar los recursos de manera definitiva para que otro agente pueda utilizarlos. Este bloque debe programarse de la siguiente manera.

- *Release: All seized resources from given pool(s).* De este modo se liberan todos los recursos de los conjuntos de recursos que se seleccionen a continuación.
- *ResourcePool objects:* Aquí se deben ir añadiendo los grupos de recursos que se quieran liberar. En este caso se seleccionan las *camas*.
- *On enter:* En este apartado se debe programar que la ocupación de las camas vaya disminuyendo. Para ello la variable *ocupacion_camas* debe actualizarse a medida que van pasando agentes por este bloque.

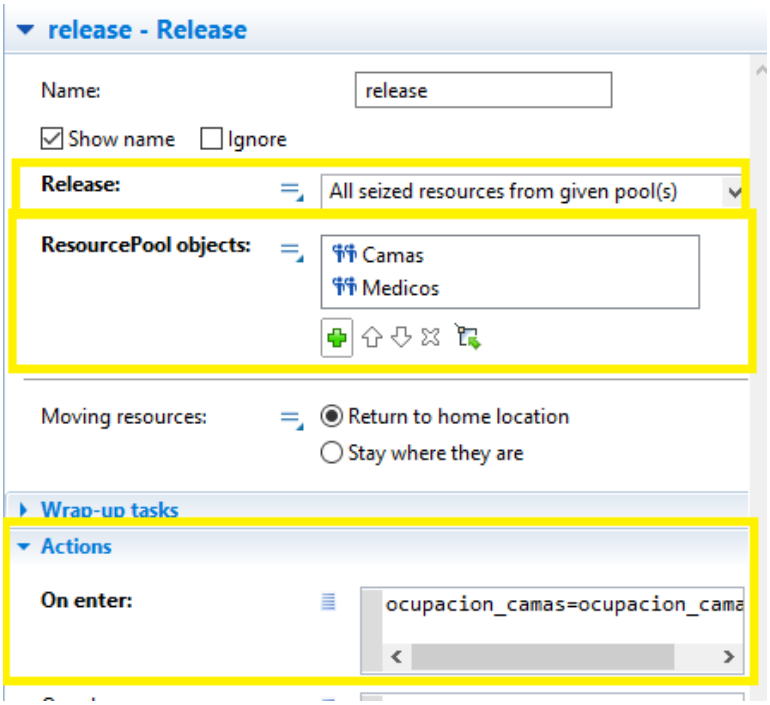


Figura 54: *Elementos a modificar en bloque Release.*

4.2.5. Altas prematuras

Para simular las altas prematuras se han empleado tres bucles: el bucle de la decisión de altas prematuras, el bucle del sorteo de las altas prematuras y el bucle del cálculo de la ocupación de las camas. A continuación se realiza una explicación detallada de cada uno de ellos.

4.2.5.1. Decisión de altas prematuras

Este bucle es el encargado de iniciar el proceso de decisión de altas prematuras. Las variables que se han utilizado en este bucle son las siguientes.

- ***pred_hoy***: esta variable almacena para cada día el número de pacientes de cirugía programada que quedan por llegar a la UCI.
- ***decison_altas_mañanas***: es una variable boleana que se emplea para activar el bucle correspondiente a la decisión de altas de las mañanas.
- ***decisión_altas_tardes***: es una variable boleana que se emplea para activar el bucle correspondiente a la decisión de altas de las tardes.

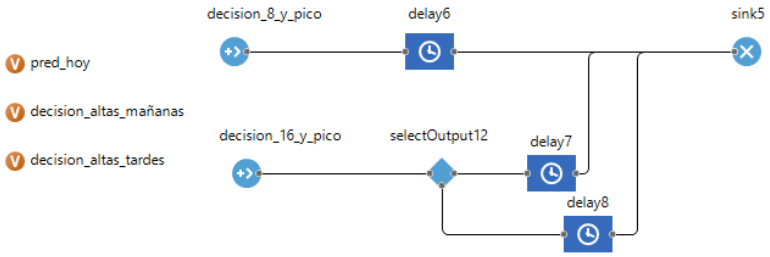


Figura 55: *Bucle que representa las reuniones entre médicos para la decisión de altas.*

Por lo tanto, el lazo se inicia con dos bloques *source* que generan una entidad al día uno a las 08:20 de la mañana y el otro a las 16:20 de la tarde, simulando las reuniones que tienen los médicos para dar altas. Estos horarios se pueden modificar sencillamente si se desea que las reuniones tengan lugar en otra franja horaria. En los bloques, debe programarse lo siguiente:

- *Interarrival time*: debe establecerse constante y de 1 día para que se realice una vez al día.
- *Time of start*: hace referencia al instante en el que se crea la primera entidad. Al poner 500 y 980 minutos se está asumiendo que la primera entidad se crea a las 08:20 de la mañana y 16:20 de la tarde respectivamente. Este tiempo puede modificarse en función de a qué hora se desee realizar la reunión.
- *On exit*: cuando la entidad sale del bloque, hay que indicarle que active la variable de decisión de mañana o de la tarde.

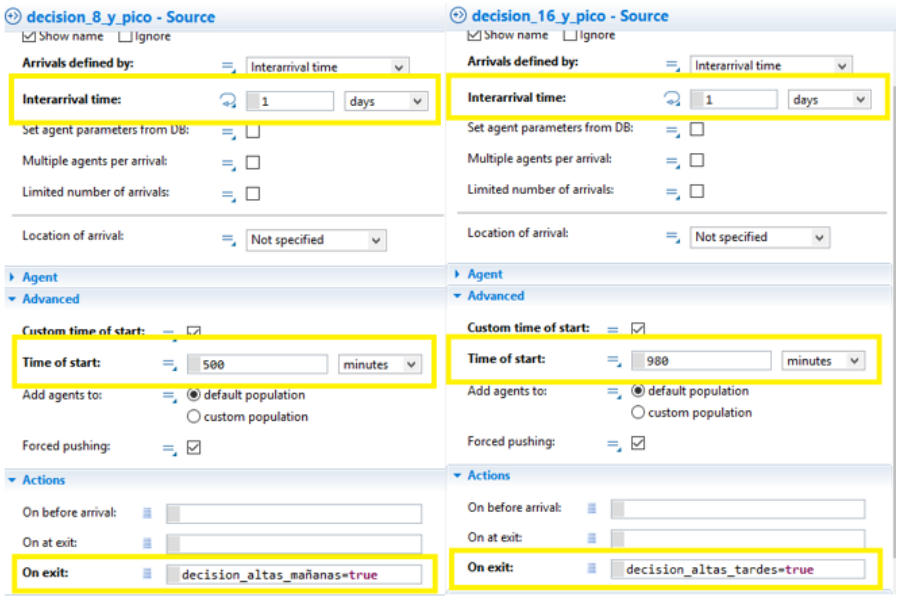


Figura 56: *Elementos a modificar en bloque Source del bucle de decisión de altas prematuras.*

Posteriormente, estas entidades generadas pasan a sus correspondientes bloques *delay*. Cabe destacar que el tiempo de espera de estos bloques es despreciable debido a que se han situado únicamente con el fin de aprovechar el apartado *Actions* de cada uno de ellos para realizar distintos cambios en las variables.

La programación que se debe hacer en estos bloques está relacionada con la variable *pred_hoy*. Cuando se activa la decisión de las 08:20 de la mañana, todavía no se ha producido ninguna llegada de pacientes programados por lo que el valor de la variable coincide con el primer componente del vector de predicciones (*pred_hoy=pred[0]*). Sin embargo, cuando se inicia el proceso a las 16:20 de la tarde, es posible que haya llegado un paciente (dependiendo del valor que corresponda para ese dia) por lo que el valor de la variable es distinto. Por lo tanto se debe asignar dicho valor en función de cual sea el número de llegadas totales en el día.

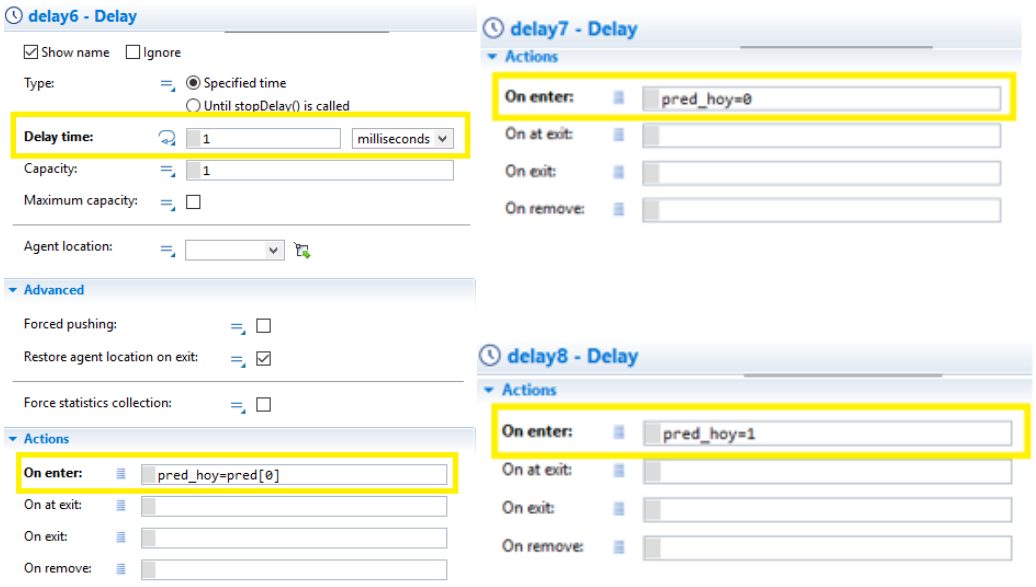


Figura 57: *Elementos a modificar en bloques Delay del bucle de decisión de altas prematuras.*

4.2.5.2. Cálculo de ocupación de camas

La ocupación de las camas de la UCI es un factor que influye directamente en la decisión de altas debido a que la probabilidad calculada con la que se realiza el sorteo de alta se ve influenciada por la ocupación de las camas en ese momento. Por lo tanto, se crea un lazo que contabilice en todo momento el número de camas ocupadas. Dicho lazo contiene las siguientes variables:

- ***ocupación_camas**:* esta variable muestra la ocupación real de las camas en todo momento. Crece a medida que los pacientes entran en la UCI y disminuye a medida que la abandonan.
- ***ocupación_camas_1**:* es la variable que se utiliza para actualizar el valor de la ocupación de las camas cuando se da algún alta prematura en la UCI. El valor de la variable es igual que el de la variable *ocupación_camas* pero cuando el sorteo de salida indica sacar un paciente esta variable se actualiza en función del número de pacientes que hayan salido. Una vez finalizado el bucle del sorteo de las altas vuelve a tener el mismo valor que *ocupación_camas.*
- ***ocupación_camas_virt17**:* variable boleana que es verdadera si la ocupación de las camas está entre 0 y 17.
- ***ocupación_camas_virt18***: variable boleana que es verdadera si la ocupación de las camas es 18.
- ***ocupación_camas_virt19**:* variable boleana que es verdadera si la ocupación de las camas es 19.
- ***ocupación_camas_virt_20**:* variable boleana que es verdadera si la ocupación de las camas es 20.

Si se desea aumentar el número de camas se deben crear tantas variables como camas en la UCI haya y la programación es sencillamente extensible al número de camas deseado. Así pues, el bucle que realiza este cálculo de la ocupación de las camas presenta la siguiente forma:

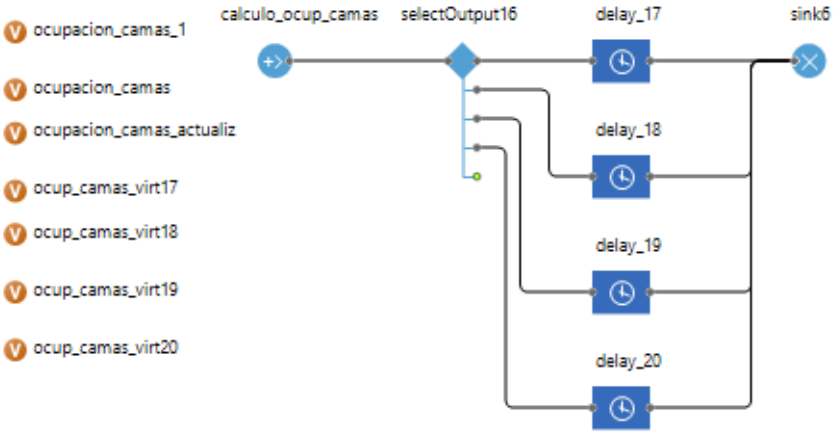


Figura 58: *Bucle correspondiente al cálculo de la ocupación de las camas de la UCI.*

El bucle se inicia con un bloque *source* que genera entidades de manera que esté constantemente actualizando el valor de la ocupación de las camas. Por otra parte, cada vez que se genere una entidad la variable *ocupación_camas_1* toma el valor de la variable *ocupación_camas,* que es la variable que se actualiza con las salidas prematuras de la UCI.

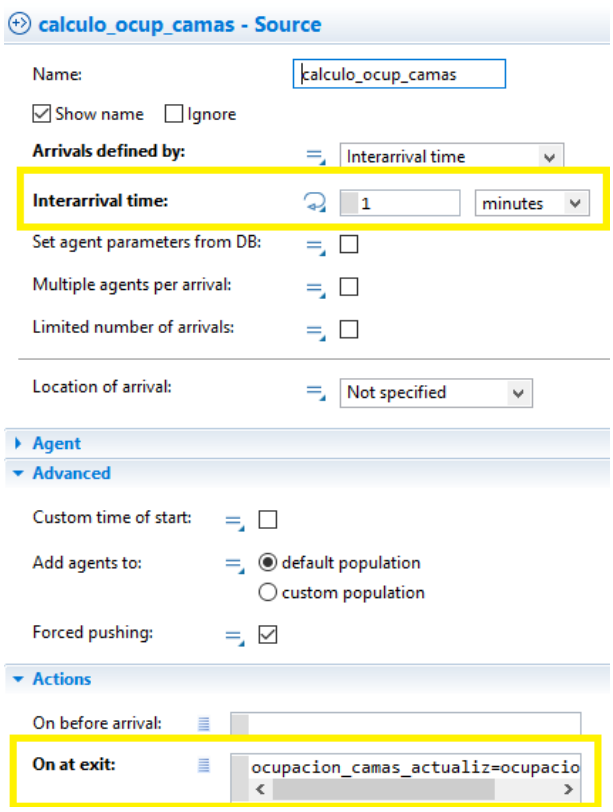


Figura 59: *Elementos a modificar en bloque Source del bucle de cálculo de ocupación de camas.*

A continuación, se programa un *selectOutput* para que en función de la *ocupación_camas_1* envíe la entidad por uno de los 4 bloques *delay* que hay situados a continuación. Cada uno de ellos se encarga de darle valor 1 a la variable que coincida con la ocupación de las camas y 0 al resto.

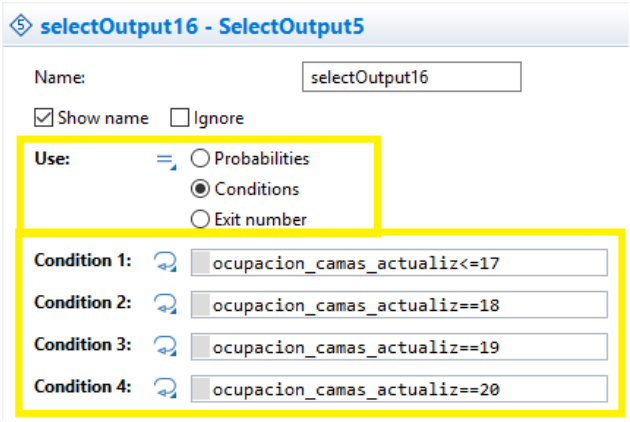


Figura 60: *Elementos a modificar en bloque Select Output del bucle de cálculo de ocupación de camas.*

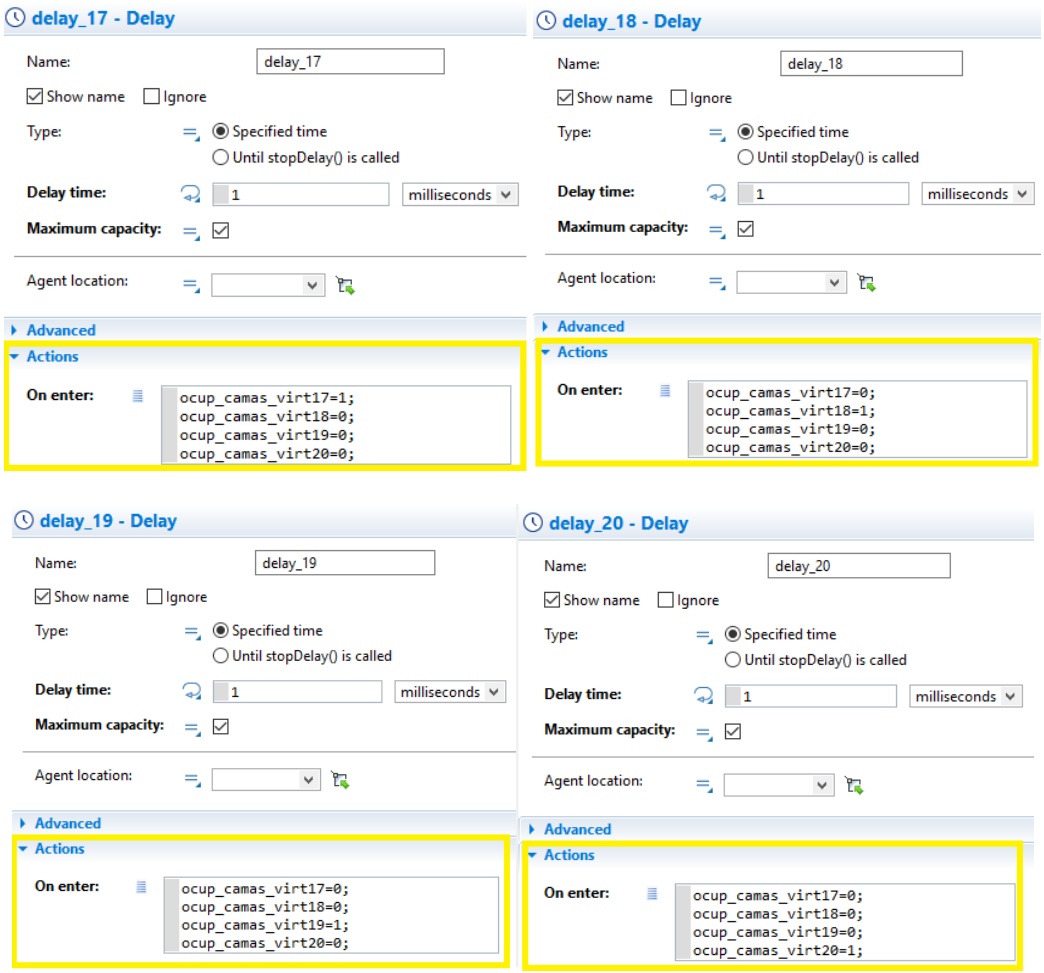


Figura 61: *Elementos a modificar en bloque Delay del bucle de cálculo de ocupación de camas.*

4.2.5.3. Sorteo de altas prematuras

Por último, se encuentra el bucle encargado de realizar el sorteo de las salidas prematuras de los pacientes. A continuación se detallan las variables que participan en el así como el funcionamiento del lazo teniendo en cuenta que el número de estados de la distribución de tipo fase mencionada en el apartado 3.4. es 5 (las condiciones de decisión del alta pueden verse en el apartado 3.5. de la memoria).

- ***Candidatos_fase_5****:* variable que contiene en cada instante el número de pacientes que hay en el estado de salud 5.
- ***Num_aleatorio_sorteo_salida****:* variable que contiene un número aleatorio según la distribución deseada.
- ***Prob_beta****:* variable que contiene el valor de la probabilidad de alta prematura.

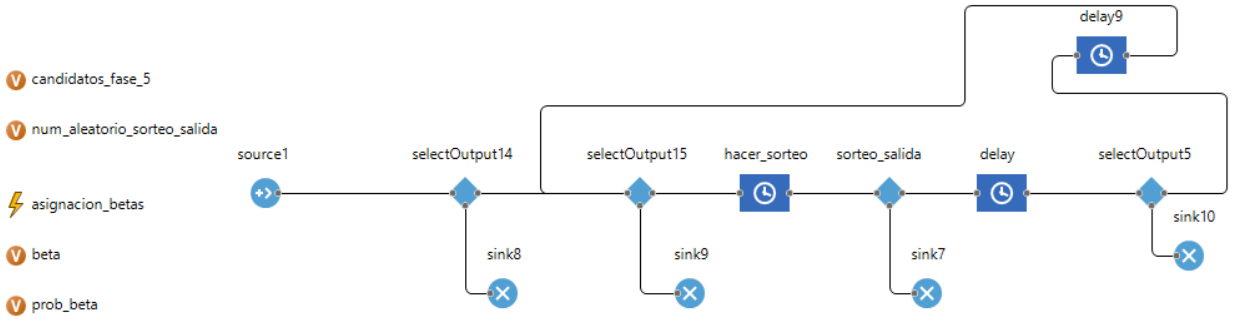


Figura 62: *Bucle correspondiente al sorteo de altas prematuras.*

El lazo se inicia con un bloque *source* que genera una entidad a la hora. Estas entidades pasan a un bloque del tipo *selectOutput* que tiene la siguiente condición: *decision_altas_mañanas==true || decision_altas_tardes==true.* Está programado de esta manera para que en el momento que una de estas dos variables se active, se inicie el proceso del sorteo de altas. Estas variables se activan en el bucle de decisión de altas prematuras que se ha explicado previamente. Con esto se consigue que este bucle se inicie únicamente cuando sea necesario dar un alta prematura.

Por otra parte, en el apartado de acciones se debe programar que la variable *ocupacion_camas_1* tome el mismo valor que el de *ocupacion_camas.*

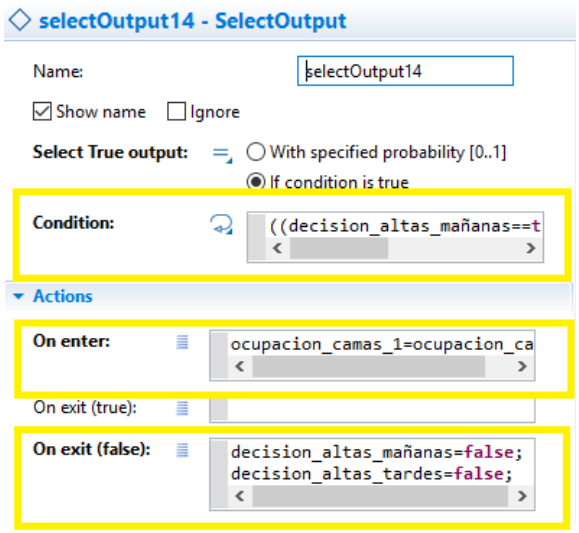


Figura 63: *Elementos a modificar en bloque SelectOutput14 del bucle de sorteo de altas prematuras.*

Una vez que se inicia este proceso, las entidades pasan a otro bloque similar cuya condición para avanzar es que haya algún candidato en estado 5 de salud. Además, se vuelven a poner en *false* las dos variables encargadas de iniciar el proceso.

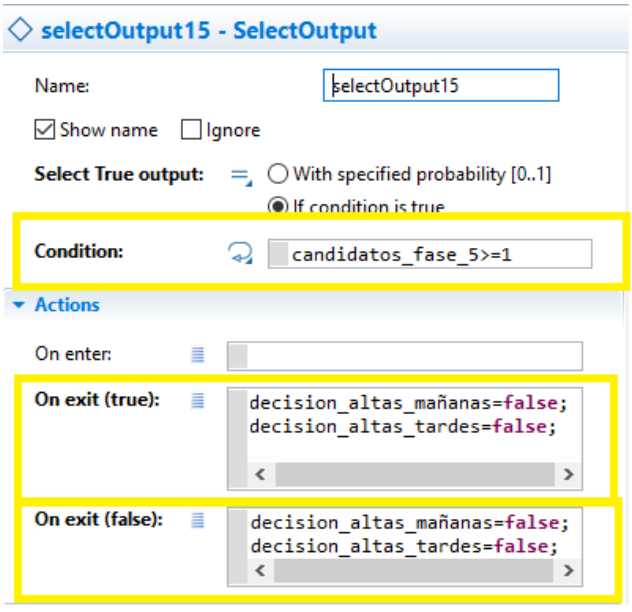


Figura 64: *Elementos a modificar en bloque SelectOutput15 del bucle de sorteo de altas prematuras.*

Por lo tanto, si hay algún candidato en el estado 5 de salud la entidad avanza hasta el siguiente bloque que es de tipo *delay.* Aquí se realiza el cálculo de la probabilidad de alta prematura y se almacena en la variable *prob_beta.*

El siguiente paso es realizar el sorteo. Para ello, se emplea otro bloque *selectOutput* que compara el número aleatorio generado con la probabilidad de alta prematura. Si *num_aleatorio_sorteo_salida* ≤ *prob_beta* el sorteo indica que se debe acortar la estancia de un paciente y en el caso contrario ha finalizado el proceso de decisión de alta prematura.

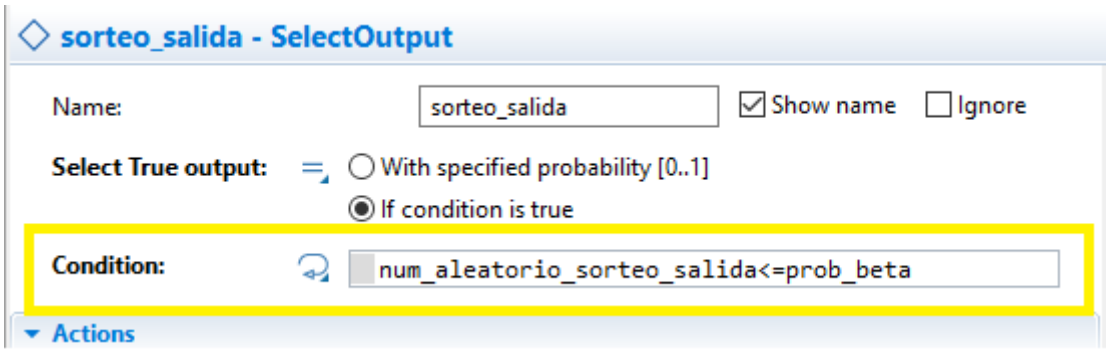


Figura 65: *Elementos a modificar en bloque sorteo_salida del bucle de sorteo de altas prematuras.*

En caso de que el sorteo diga que se debe acortar la estancia de un paciente la entidad avanza hasta el siguiente bloque, que es el que contiene la programación necesaria para que esto ocurra.

```
for (int j=0; j<1; j++) {

paciente t = null;
if(estado5.size()>0){

    //use a random agent stored at estado5
    int i = uniform_discr(0, estado5.size()-1);
    t=estado5.get(i);
}
       t.prematuro=true;
       estado5.stopDelay(t);

}
```

Este código funciona de la siguiente manera: el bucle ***for*** está programado para que se ejecute una única vez (j=0) debido a que por cada paciente se realiza un sorteo de manera que si el sorteo dice acortar estancia el bucle únicamente permite acortar la estancia de un paciente. Se establece la variable ***t***, que va a hacer referencia a los agentes de tipo ***paciente***.

Por lo tanto, mediante un bucle ***if*** se programa que si hay agentes en el estado 5 de salud se selecciona uno de manera aleatoria y se le asigna a la variable *t*. Una vez que se tiene el agente cuya estancia va a ser acortada, se emplea el comando ***stopDelay(t)*** para dar por finalizado su tiempo de retardo en el estado absorbente y sale del bloque. Además, se indica que el parámetro ***prematuro*** de ese paciente se active (true) ya que más adelante es necesario distinguir este tipo de pacientes del resto.

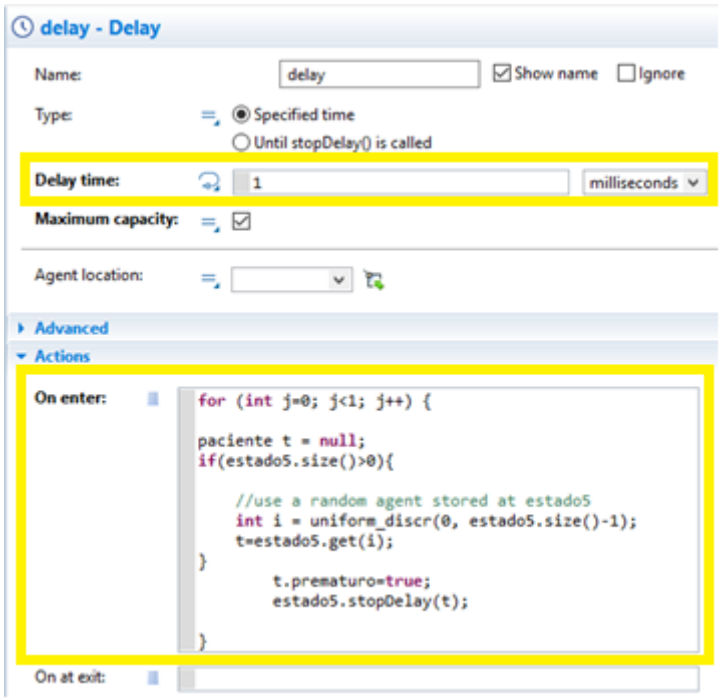


Figura 66: *Elementos a modificar en bloque Delay del bucle de sorteo de altas prematuras.*

Posteriormente, la entidad pasa a un bloque *selectOutput* con probabilidad 1 de manera que siempre pasa al siguiente bloque, que es un bloque tipo *delay* con un tiempo de retardo despreciable pero que permite programar otra condición.

En este caso, se debe programar que la ocupación de las camas ha disminuido ya que como se va a realizar otro sorteo para acortar la estancia de otro paciente, la

ocupación de las camas debe ser actualizada y va a afectar al cálculo de la probabilidad. Para ello se utiliza la variable *ocupación_camas_1* que es la variable que tiene en cuenta el bucle del cálculo de la ocupación de las camas.

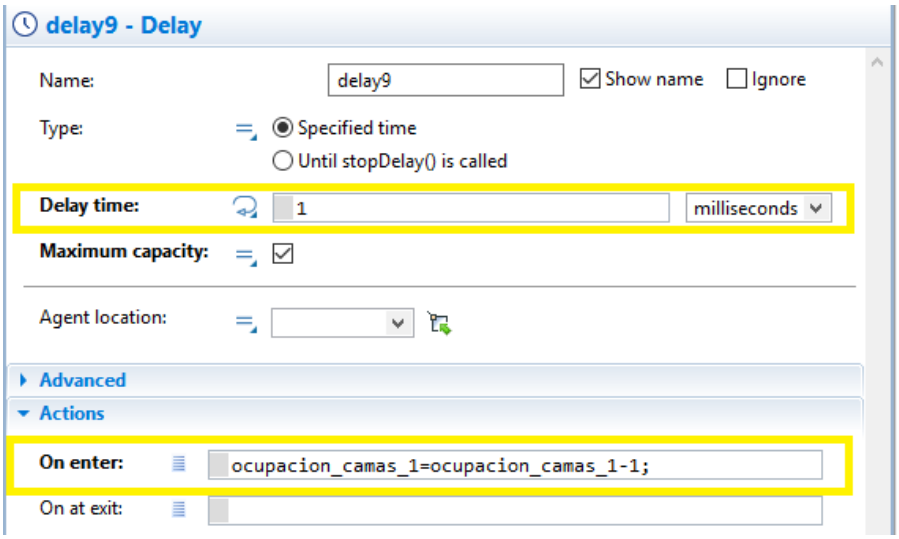


Figura 67: *Elementos a modificar en bloque delay9 del bucle de sorteo de altas prematuras.*

Este proceso se ejecuta hasta que el sorteo diga que no se acorta la estancia de los pacientes. En ese momento finaliza el sorteo de las altas prematuras.

5. EJEMPLO ILUSTRATIVO Y RESULTADOS

Con el objetivo de ilustrar el funcionamiento del modelo de simulación de UCI propuesto, se ha particularizado este modelo para un escenario concreto, basado en una UCI real de 20 camas, analizado en [3].

A continuación se procede a explicar los valores de los parámetros del ejemplo y manera de implementar en Anylogic dicho modelo.

5.1. Implementación del ejemplo ilustrativo

5.1.1. Proceso de llegadas

Como se ha comentado anteriormente, para en este modelo se han resumido los pacientes en dos grupos: pacientes grupo 1 y 2. La manera en la que se producen las llegadas para cada tipo de paciente es la siguiente.

- **Grupo de pacientes 1: llegadas no programadas (pacientes de grupo 1)**

Los pacientes de este tipo llegan de acuerdo a un Proceso de Poisson con tasa λ=1,11 pacientes/día. Las llegadas se producen las 24 horas del día durante los 7 días de la semana.

- **Grupo de pacientes 2: llegadas programadas (pacientes de grupo 2)**

Estos son los pacientes que llegan tras cirugías programadas, y cuya llegada se conoce con antelación. Las llegadas programadas se producen de lunes a viernes de acuerdo a la siguiente distribución de probabilidad: llegan 2 pacientes con probabilidad de 0.2, 1 paciente con probabilidad de 0.6 y ningún paciente con probabilidad de 0.2. Se asume que en el supuesto de que llegue un único paciente su hora de entrada a la UCI es las 12h de la mañana y si llegan dos pacientes uno entra a las 12h y el segundo a las 20h. Por otra parte, se asume que el personal médico conoce la previsión de la llegada de pacientes programados en un horizonte 3 días de antelación.

Número de pacientes	Probabilidad de llegada
0	0.2
1	0.6
2	0.2

Tabla 2: *Tabla de probabilidades para llegadas programadas.*

Por lo tanto, en el apartado del bucle donde se realiza la predicción de las llegadas se debe programar lo siguiente. Este bucle, es el que se ha detallado en el apartado 4.2.2. (Figura 37).

Si el día de la semana es menor que 5 quiere decir que es de lunes a viernes y por lo tanto se debe realizar el sorteo del número de llegadas. Para ello, se emplea un bloque *selectOutput5* y se programa que si el número aleatorio generado anteriormente está entre 0-0.2 se asigna un valor de 0 (*pred[6]=0)* a las llegadas de ese día, si está entre 0.2-0.8 se asigna un valor de 1 llegadas y si está entre 0.8-1 se asigna un valor de 2.

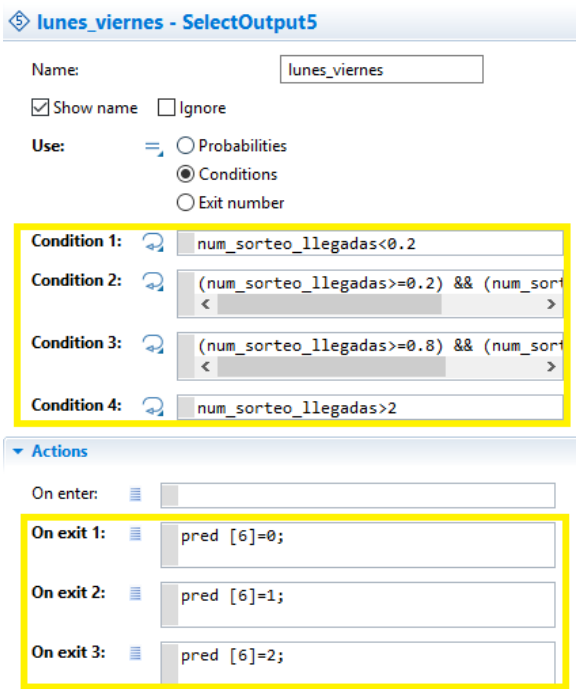


Figura 68: *Elementos a modificar en bloque lunes_viernes del bucle de predicción de llegadas programadas.*

Por otra parte, si el día de la semana es mayor que 5 quiere decir que es fin de semana. Si es sábado (dia_sem==6) se programan 0 llegadas y si es domingo se programan 0 llegadas y se pone el contador del día de la semana a cero.

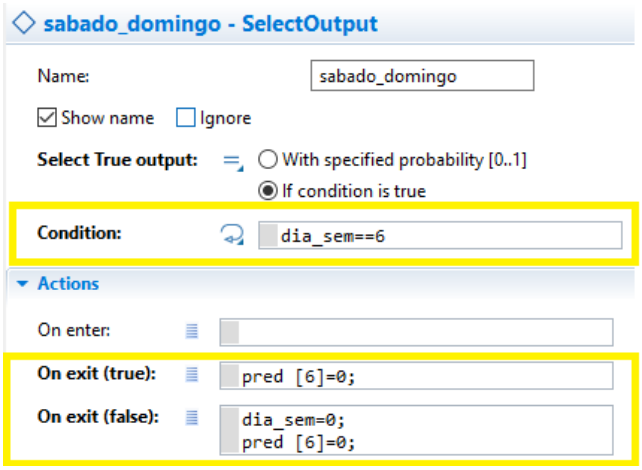


Figura 69: *Elementos a modificar en bloque sabado_domingo del bucle de predicción de llegadas programadas.*

5.1.2. Estancia de los pacientes

Como se ha comentado en el apartado 3.4. de la memoria una de las distribuciones más adecuadas para modelar los tiempos de estancia de los pacientes son las distribuciones tipo fase. El artículo mencionado propone una distribución de tipo fase con cinco estados distintos. Cada tipo de paciente tiene unos tiempos medios de estancia y una matriz de transiciones entre estados. Por ejemplo, los tiempos medios de estancia para cada uno de los estados de los pacientes de grupo 1 no exitus son exponenciales de 1, 6, 2, 2 y 2 días. En la imagen a continuación se muestra su distribución tipo fase.

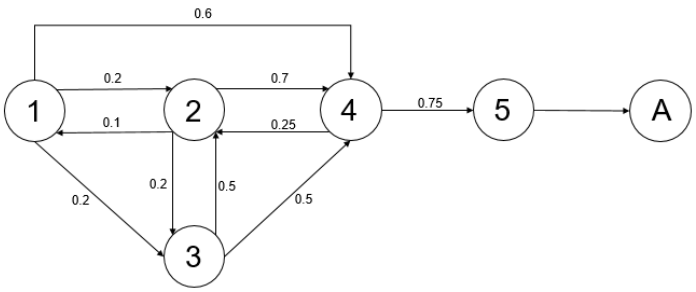


Figura 70: *Distribución de tipo fase para modelar el tiempo de estancia de los pacientes.*

Además, las transiciones entre un estado y otro vienen dadas por distintas probabilidades para cada tipo de paciente. Dichas probabilidades y los tiempos medios de estancia (m_i) para cada estado se muestran a continuación.

Pacientes grupo 1 y no exitus

$$\begin{pmatrix} 0 & 0.2 & 0.2 & 0.6 & 0 \\ 0.1 & 0 & 0.2 & 0.7 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0.25 & 0 & 0 & 0.75 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

m_1=1, m_2=6, m_3=2, m_4=2, m_5=2

Pacientes grupo 1 y exitus

$$\begin{pmatrix} 0 & 0.45 & 0 & 0 & 0 \\ 0.65 & 0 & 0.35 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

m_1=1, m_2=2, m_3=1

Pacientes grupo 2 y no exitus

$$\begin{pmatrix} 0 & 0.15 & 0 & 0.85 & 0 \\ 0 & 0 & 0.15 & 0.85 & 0 \\ 0 & 0.15 & 0 & 0.85 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

m_1=1, m_2=6, m_3=2, m_4=1, m_5=1

Pacientes grupo 2 y exitus

$$\begin{pmatrix} 0 & 0.4 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$m_1=7$, $m_2=6$, $m_3=6$

Estas probabilidades deben introducirse en los correspondientes bloques *SelecOutput* que modelan la transición entre un estado y otro tal y como se explica en el apartado 4.2.3.

5.1.3. Proceso de altas

Como indica el artículo, los instantes del día en los que los médicos toman las decisiones sobre el alta de pacientes son las 08:00 de la mañana y las 16:00 de la tarde. Los pacientes que alcanzan el estado de salud absorbente entre las 20:00 de la tarde y las 08:00 de la mañana son dados de alta a las 08:00 de la mañana y los pacientes que alcanzan el estado absorbente entre las 08:00 y las 16:00 se dan de alta a las 16:00.

Por lo tanto, mediante un bloque *selectOutput* se programa para que si la hora en la que el agente pasa por el bloque está entre las 8h y las 16h sea enviado al bloque de salida *tardes* y en el resto de casos es enviado al bloque de salida *mañanas.*

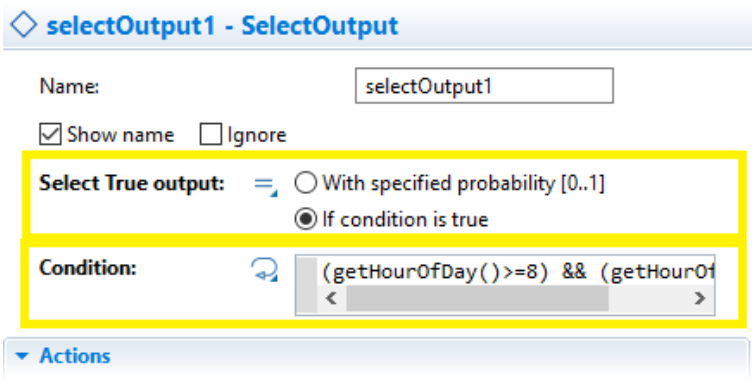


Figura 71: *Elementos a modificar en bloque selectOutput11 del bucle de salida de los pacientes.*

Como se ha comentado en el apartado 4.2.4. se deben programar los eventos que ejecuten la toma de decisiones de alta de pacientes. Por lo tanto, se programan para que los eventos ocurran a las 8h de la mañana y a las 16h de la tarde.

Por otra parte, se debe asignar un valor a los tiempos SetupTime1 y SetupTime2 mencionados en el apartado 4.2.4. El tiempo SetupTime1, correspondiente al tiempo de tramitación del alta de un paciente, se aproxima con una distribución Uniforme(2,4) en horas y el tiempo SetupTime2, correspondiente al tiempo de limpieza y preparación de la cama para el siguiente paciente, se aproxima con una distribución Uniforme (1,2), en horas.

5.1.4. Altas prematuras

5.1.4.1. Función de probabilidad

Como se ha comentado anteriormente en la memoria, para modelar las altas prematuras se considera una función de tipo probabilístico (ver detalles en [3]) que tiene en cuenta tres elementos: la ocupación de las camas *(y)* en ese momento, el estado de salud del paciente *(k*) y el número de llegadas de pacientes de cirugía programada (***X)*** previstos para los próximos días. De este modo, se define la probabilidad de alta prematura como la probabilidad de alta para un paciente en estado $k \in DS$ (siendo DS el conjunto de estados en los que un paciente podría ser dado de alta de la UCI) cuando hay *y* camas ocupadas y con una previsión para los próximos días de llegadas programadas descrita por el vector ***X***. Se propone la siguiente función logística para ~~el~~ el modelado de la probabilidad de alta prematura.

$$Pk(y, X) = \frac{1}{1 + \exp(-\beta(1, Z)')}$$

donde ***β*** es un vector de coeficientes cuyo cálculo se detalla más adelante y ***Z*** es el vector de predictores de la función logística que depende de la ocupación de las camas (***y)*** y de la previsión de las llegadas programadas de cirugía en un horizonte de *h=3* días ***X*** = $(x_1,\ldots,x_h)$.

Siguiendo [referencia artículo] se define *β(1,Z)'* como:

$$\beta(1,Z)' = \beta_0 + \sum_{i=1}^{3}\sum_{n=18}^{20} \beta_{in} \cdot Z_{in} =$$

$$= \beta_0 + \beta_1 X_1 1_{\{y=20\}} + \beta_2 X_1 1_{\{y=19\}} + \beta_3 X_1 1_{\{y=18\}} + \beta_4 X_2 1_{\{y=20\}}$$
$$+ \beta_5 X_2 1_{\{y=19\}} + \beta_6 X_2 1_{\{y=18\}} + \beta_7 X_3 1_{\{y=20\}} + \beta_8 X_3 1_{\{y=19\}}$$
$$+ \beta_9 X_3 1_{\{y=18\}}$$

Cálculo de coeficientes de la función de probabilidad

Para calcular los coeficientes $\beta = (\beta_0, \ldots, \beta_9)$ se formula y resuelve un problema de optimización, en el que las variables son las componentes del vector β y en el que se consideran dos funciones objetivo en conflicto: minimizar el número de pacientes rechazados en la UCI y maximizar el tiempo de estancia de los pacientes. Por lo tanto, el problema de optimización puede formularse matemáticamente de la siguiente manera:

$$\underset{\beta}{Min} \quad P_R$$

$$\underset{\beta}{Max} \quad E[t_{estancia}]$$

$$Sujeto\ a \quad \begin{cases} p_k(y,X) = \frac{1}{1+\exp(-\beta(1,Z)')} \\ \beta \in \mathbb{R}^n \end{cases}$$

Para la resolución de este problema de optimización multiobjetivo se utiliza al método de las ε-restricciones:

$$P[\varepsilon] \qquad \underset{\beta}{Max} \quad E[t_{estancia}]$$

$$Sujeto\ a \quad \begin{cases} p_k(y,X) = \frac{1}{1+\exp(-\beta(1,Z)')} \\ P_R \leq \varepsilon \\ \beta \in \mathbb{R}^n \end{cases}$$

El problema de optimización anterior se resuelve combinando optimización con simulación (ver detalles en [3]), obteniéndose los siguientes resultados para tres distintos porcentajes de pacientes rechazados:

β	Porcentaje de rechazo de pacientes		
	ε=6%	ε=4.5%	ε=3%
β_0	-17.2	-10	-10
β_1	7.45	8.65	18.4
β_2	4.65	6.45	18.4
β_3	0	3.8	18.4
β_4	7.45	6.45	18.4
β_5	2.3	6.45	18.4
β_6	0.25	2.65	18.4
β_7	0.25	6.45	15.55
β_8	0.2	0.5	12.75
β_9	0	0.5	4.1
E [$t_{estancia}$]	**9.21**	**9.08**	**8.86**

Tabla 3: *Valores de los coeficientes β para distintos porcentaje de rechazo de pacientes.*

5.1.4.2. Calculo de la ocupación de camas

La UCI con la que trabaja el artículo mencionado tiene 20 camas por lo que la programación de este apartado es exactamente igual que el apartado 4.2.5.2.

5.1.4.3. Sorteo de altas prematuras

A continuación se debe particularizar el bucle donde simula el procedimiento para la toma de decisiones en relación con las altas prematuras. Las variables para su programación son las siguientes.

- ***Candidatos_fase_5****:* variable que contiene en cada instante el número de pacientes que hay en el estado de salud DS, que en este modelo es DS={5}.
- ***Num_aleatorio_sorteo_salida****:* variable que contiene un número aleatorio según la distribución Unif (0,1).
- ***Prob_beta****:* variable que contiene el valor de la probabilidad de alta prematura descrita en el apartado 5.1.4.1.
- ***Beta****:* vector de variables que contiene los distintos valores de las β empleadas para el cálculo de la probabilidad de alta prematura.

Cabe destacar que para realizar este cálculo es necesario introducir cuales son los valores de las β por lo que se ha creado un evento (*asignación_betas*) que ocurre una única vez y que en el instante que se inicia la simulación asigna dichos valores. Por otra parte, se asigna el valor del número aleatorio uniforme (0,1) a la variable *num_aleatorio_sorteo_salida.*

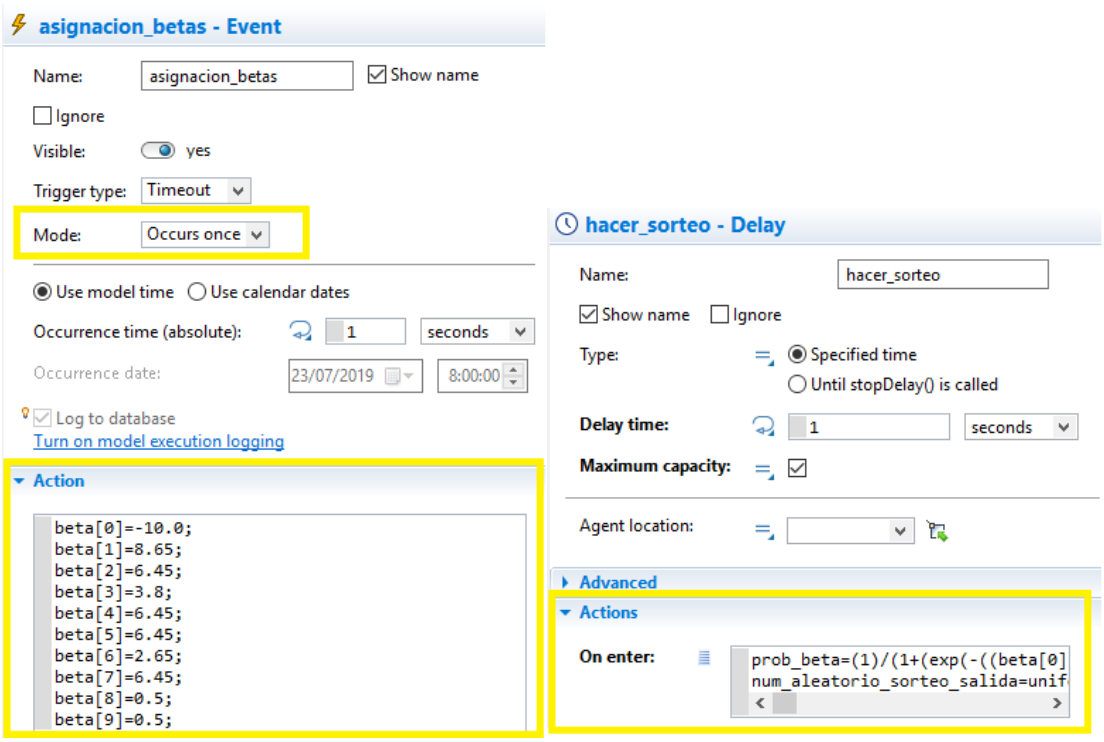


Figura 72: *Elementos a modificar en evento asignación_betas y hacer_sorteo del bucle de altas prematuras.*

5.2. Medidas de funcionamiento

En este apartado se van a detallar cuales son las medidas de funcionamiento que se quieren evaluar vía simulación. Para poder obtener estos resultados se deben realizar pequeñas modificaciones en alguno de los bucles y se deben añadir una serie de elementos. Cabe destacar que estas modificaciones no afectan en nada a todo lo mencionado anteriormente.

Las medidas de funcionamiento que se van a evaluar son las siguientes:

a) Porcentaje de pacientes rechazados

La primera medida de funcionamiento a evaluar es el porcentaje de pacientes que son rechazados en la UCI. Este resultado va a obtenerse para cada tipo de paciente.

b) Tiempos de estancia de los pacientes

La siguiente medida de funcionamiento a obtener es el tiempo de estancia de los pacientes en la UCI. En este caso se va a obtener para cada tipo de paciente, es decir, paciente de grupo 1 y grupo 2, exitus y no exitus.

c) Porcentaje de pacientes dados de alta prematuramente

Otra medida de funcionamiento a evaluar es el porcentaje de pacientes prematuros que hay en la UCI. Estos pacientes son aquellos que son dados de alta antes de llegar a la fase absorbente. Este resultado va a obtenerse para cada tipo de paciente.

d) Tiempo medio acortado

La siguiente medida de funcionamiento a obtener es el tiempo medio acortado de pacientes prematuros en la UCI. En este caso se va a obtener para cada tipo de paciente, es decir, paciente de grupo 1 y grupo 2.

e) Frecuencia de ocupación de las camas

Por último, se obtiene la frecuencia de ocupación de las camas que es la frecuencia con la que la UCI ha tenido ocupadas i camas, para i=0, 1, 2,…,20.

A continuación se muestran los resultados que van a obtenerse de la simulación, explicando que modificaciones se deben realizar para obtenerlos.

5.2.1. Porcentaje de pacientes de grupo 1 y grupo 2 rechazados

Para poder calcular cual es el porcentaje de pacientes rechazados de grupo 1 y de grupo 2 es necesario contabilizar cuantos de los pacientes que se rechazan son de cada tipo. Esto se va a realizar gracias a las siguientes variables:

- ***NumInSys_TOTAL**:* almacena el número total de pacientes que llegan a la UCI.
- ***NumInSys_G1**:* almacena el número de pacientes de grupo 1 que llegan a la UCI.
- ***NumInSys_G2**:* almacena el número de pacientes de grupo 2 que llegan a la UCI.
- ***NumInSys_G1_R**:* almacena el número de pacientes de grupo 1 que son rechazados en la UCI.
- ***NumInSys_G2_R**:* almacena el número de pacientes de grupo 2 que son rechazados en la UCI.

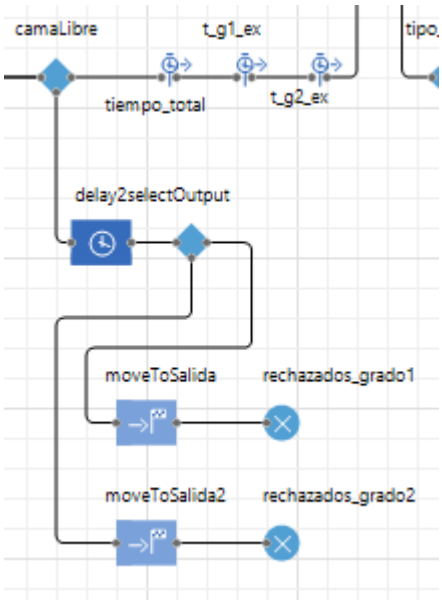


***Figura 73:** Bucle para contabilizar el número de pacientes rechazados.*

Por lo tanto, para contabilizarlos adecuadamente se ha diseñado el lazo mostrado en la Figura 73 que funciona de la siguiente manera. Los agentes que son rechazados de la UCI van a un bloque *selectOutput* y dependiendo del atributo que tengan (grupo 1 o grupo 2) son enviados a un bloque *sink.*

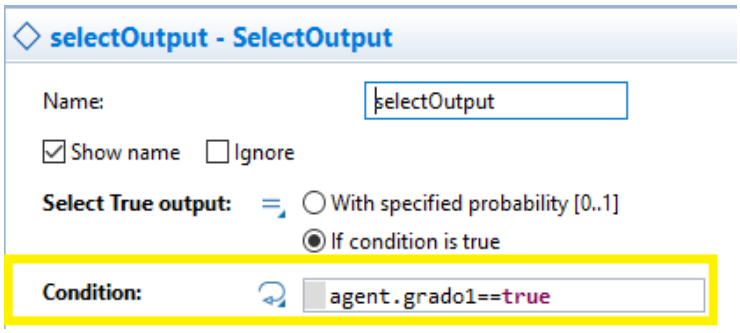


Figura 74: *Programación del bloque selectOutput del bucle para contabilizar pacientes rechazados.*

Estos bloques *sink* están programados para que cada vez que un agente pase por ellos vayan añadiendo a las variables mencionadas anteriormente. Además, en los bloques *source* donde se crean los dos tipos de agentes se debe programar que cada vez que se cree un agente se vaya contabilizando en sus correspondientes variables.

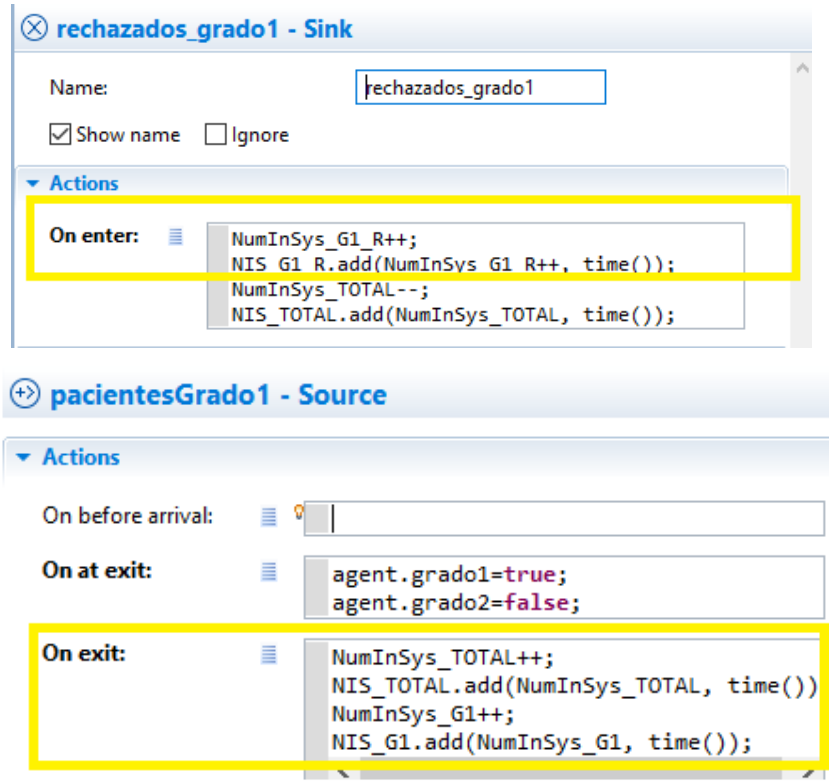


Figura 75: *Programación del bloques Sink del bucle para contabilizar pacientes rechazados.*

De esta manera, cada variable almacena los datos necesarios para poder calcular estos porcentajes de pacientes rechazados.

5.2.2. Tiempo de estancia de los pacientes de grupo 1 y grupo 2.

Otra de las medidas de funcionamiento que se deben extraer del modelo es la media del tiempo de estancia de los pacientes. En este caso, se van a distinguir los tiempos de estancia para los pacientes de grupo 1 y grupo 2. Además, se puede medir de dos maneras distintas; la primera, teniendo en cuenta únicamente el tiempo que tardan los pacientes en alcanzar el estado absorbente de salud y la segunda, teniendo en cuenta el tiempo que tardan hasta que se dan de alta y abandonan la UCI.

Para obtener estos tiempos, se van a emplear bloques del tipo *time measure start* y *time measure end.* Se deben colocar cuatro bloques *time measure start* en el instante en el que entran a la UCI, uno para cada tipo de paciente exitus y no exitus (*t_g1, t_g1_ex, t_g2* y *t_g2_ex).*

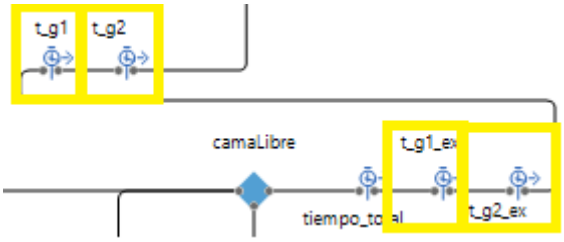


Figura 76: *Situación de bloques Time Measure Start para medición de tiempos.*

Por lo tanto, se debe colocar un bloque *time measure end* por cada uno de los bloques *time measure start* que se han puesto en el modelo. De este modo se sitúan en su correspondiente lazo, antes o después del bloque *delay* que contiene el *setupTime* (tiempo de alta del paciente), según el tiempo que se desee obtener.

Por otra parte, se puede observar que se ha diseñado un pequeño lazo a la salida del estado 5 de los pacientes que se encarga de dividir los pacientes según su tipo (grupo 1 y grupo 2) y según si han sido pacientes prematuros o no. Esto se ha realizado para poder situar los bloques de medición de tiempos de manera más sencilla y ordenada.

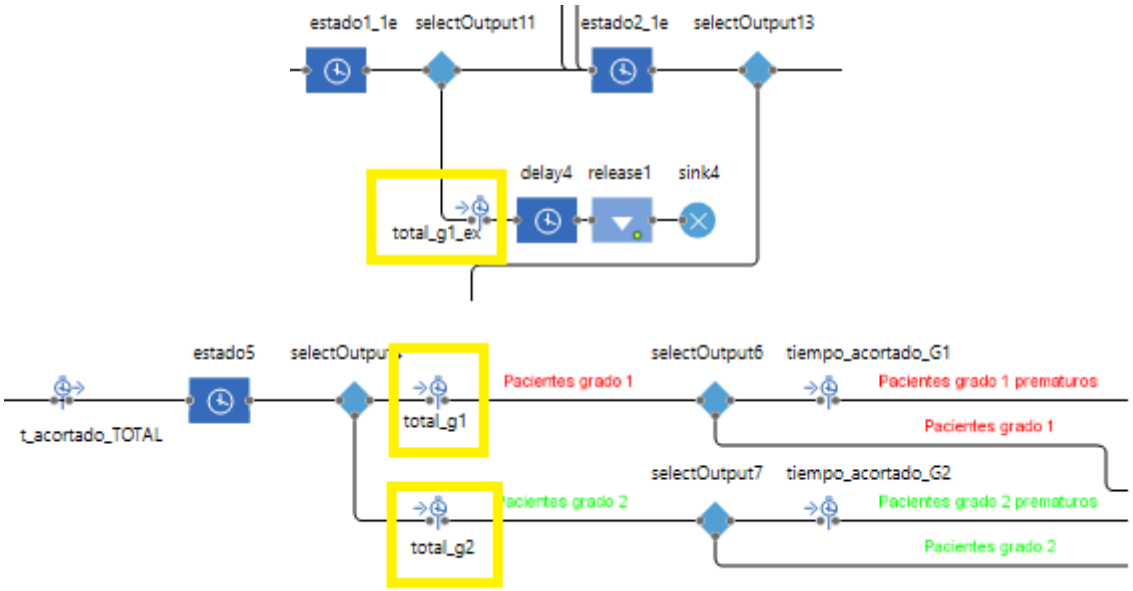


Figura 77: *Colocación de bloques Time Measure End para medición de tiempos.*

5.2.3. Porcentaje de pacientes dados de alta prematuramente de grupo 1 y de grupo 2

Para poder calcular cual es el porcentaje de pacientes dados de alta prematuramente de grupo 1 y de grupo 2 es necesario contabilizar el número exacto de pacientes que se dan de alta antes de llegar al estado de salud absorbente y almacenarlo en distintas variables. Las variables son las siguientes:

- ***NumInSys_G1_P****:* almacena el número de pacientes de grupo 1 que se dan de alta antes de tiempo.
- ***NumInSys_G2_P****:* almacena el número de pacientes de grupo 2 que se dan de alta antes de tiempo.

Por lo tanto, para contabilizarlos adecuadamente se aprovecha el lazo que se que se ha diseñado que distingue los pacientes dependiendo de si son de pacientes dados de alta prematuramente o no. Para ello, en la correspondiente salida del bloque *selectOutput* se programa para que a medida que vayan pasando agentes se vaya sumando en la variable.

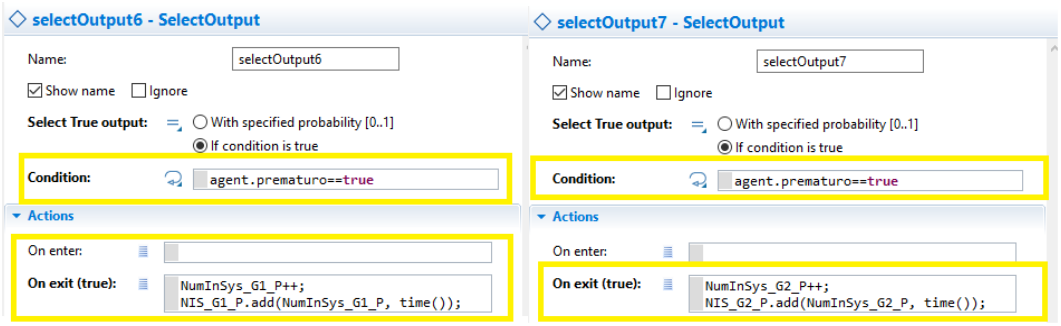


Figura 78: *Programación de bloques SelectOutput para separar y contabilizar pacientes prematuros.*

De este modo, cada una de estas variables almacena los datos necesarios para poder calcular estos porcentajes de pacientes cuya estancia ha sido acortada.

5.2.4. Tiempo medio acortado: total y para los pacientes de grupo 1 y grupo 2

Una medida de funcionamiento importante que se debe conocer tras realizar la simulación de este modelo es el tiempo que se acorta la estancia de los pacientes tanto para los pacientes de grado de urgencia 1 como para los pacientes de grado de urgencia 2.

Para calcular estos tiempos, al igual que se ha realizado anteriormente, se van a utilizar bloques del tipo *time measure start/end.* Teniendo en cuenta que se conoce cuál es el tiempo de estancia en cada uno de los estados basta con distribuir estos bloques de la siguiente manera.

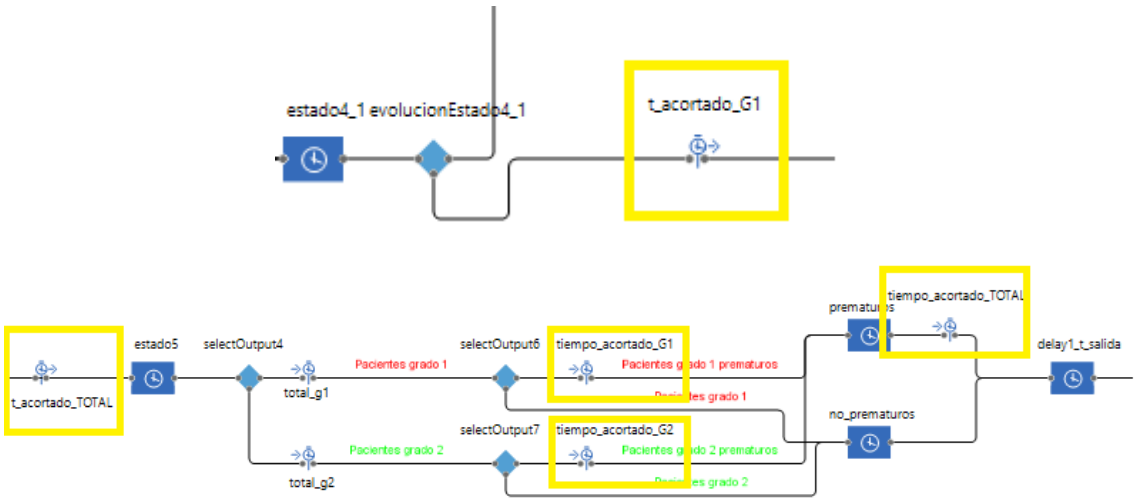


Figura 79: *Colocación de bloques Time Measure Start/End para medición de tiempos acortados.*

Se deben colocar 3 bloques de *time measure start* ya que se quieren obtener 3 tiempos distintos; el tiempo total acortado, el tiempo acortado para los pacientes de grupo 1 y el tiempo acortado para los pacientes de grupo 2. Se va a aprovechar el lazo que se ha creado para dividir los pacientes según su tipo.

Por lo tanto, a la salida del estado 4 de cada tipo de paciente se colocan dos bloques con nombre *t_acortado_G1* y *t_acortado_G2* (Figura 79) y a la entrada del bloque del estado 5 se coloca el tercero con nombre *t_acortado_TOTAL* (Figura 79).

Una vez colocados los bloques que van a iniciar las mediciones de tiempo, se deben colocar otros 3 para finalizar cada una de estas. Los dos primeros bloques *time measure end* se colocan en las ramas del lazo correspondientes a cada tipo de pacientes prematuros (*tiempo_acortado_G1* y *tiempo_acortado_G2)* y el tercero se sitúa después del bloque *delay* con nombre *prematuros* (*tiempo_acortado_TOTAL)*.

5.2.5. Frecuencia de ocupación de camas

El último resultado de interés que se debe obtener es la distribución de probabilidad de la ocupación de camas en la UCI, es decir la frecuencia con la que la UCI ha tenido ocupadas *i* camas, para *i=0,1,2,…,20.*

Para poder obtener estas medidas de funcionamiento, se ha diseñado un lazo en el modelo que se encarga de recoger el tiempo que están ocupadas *i* camas, para cada nivel de ocupación *i,* con *i=0,1,…,20*. A continuación se explica cómo funciona el lazo y las variables que participan en él.

Las variables que participan son las siguientes:

- ***ocupacion_camas****:* variable que almacena la ocupación real de las camas de la UCI en todo momento.
- ***ocupacion_camas_antes****:* variable que almacena la ocupación de las camas en el instante anterior.
- ***tiempo_entrada_i*** (*i=0,…, 20):* almacena el instante de tiempo que entra una entidad al bloque tipo *delay i.*
- ***tiempo_i*** (*i=0,…, 20):* almacena el tiempo que ha pasado la entidad en dicho bloque.

- ***tiempo_total_i*** (*i=0,…, 20):* variable que almacena la suma de todos los tiempos que pasan las entidades en los bloques.

Como se puede observar, las 3 variables que almacenan tiempo se repiten para cada una de las ocupaciones de cama, desde 0 hasta 20.

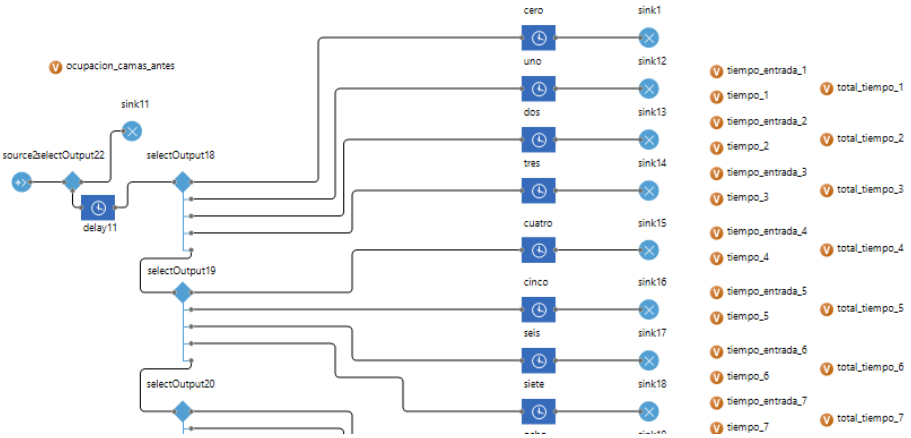


Figura 80: *Bucle correspondiente al cálculo de la frecuencia de ocupación de camas.*

Para calcular estos tiempos se emplea el siguiente lazo. Un bloque *source* genera una entidad y (Figura 80) mediante un bloque *selectOutput* se comparan las variables de *ocupacion_camas* y *ocupacion_camas_antes*; si la resta entre ambas variables es cero, la entidad desaparece del lazo por el bloque *sink* y si es distinto a cero, la entidad avanza en el lazo hacia el bloque delay, ya que esto indica que la ocupación ha cambiado. Cuando la entidad avanza en el lazo, accede a su correspondiente bloque en función de cual sea la ocupación de las camas real y donde las variables toman los tiempos que se han mencionado arriba. Estos bloques van de cero a veinte y se programan de la siguiente manera.

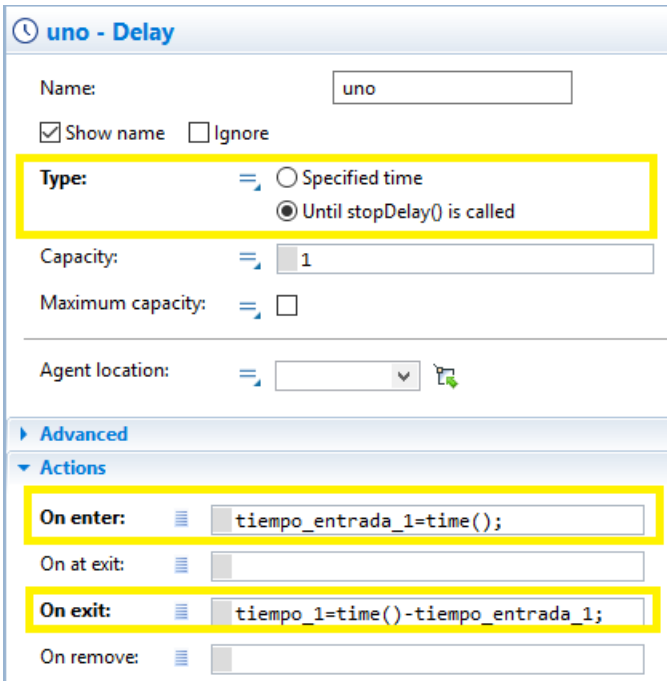
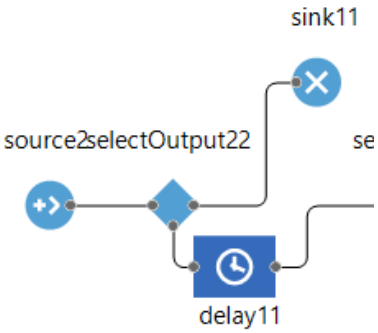


Figura 81: *Elementos a modificar en los bloques Delay del bucle del cálculo de la frecuencia de ocupación de camas.*

Por otra parte, cabe destacar que en el momento que la entidad sale por el puerto *false* del bloque *selectOutput* (Figura 81) se actualiza la variable *ocupacion_camas_antes* al valor actual de la ocupación de las camas. Por otra parte, en el bloque *delay* se ejecuta el comando *stopDelayForAll* para cada uno de los bloques *delay* que representan la ocupación ya que al haber cambiado la ocupación de las camas la entidad correspondiente debe salir del bloque donde se miden los tiempos.



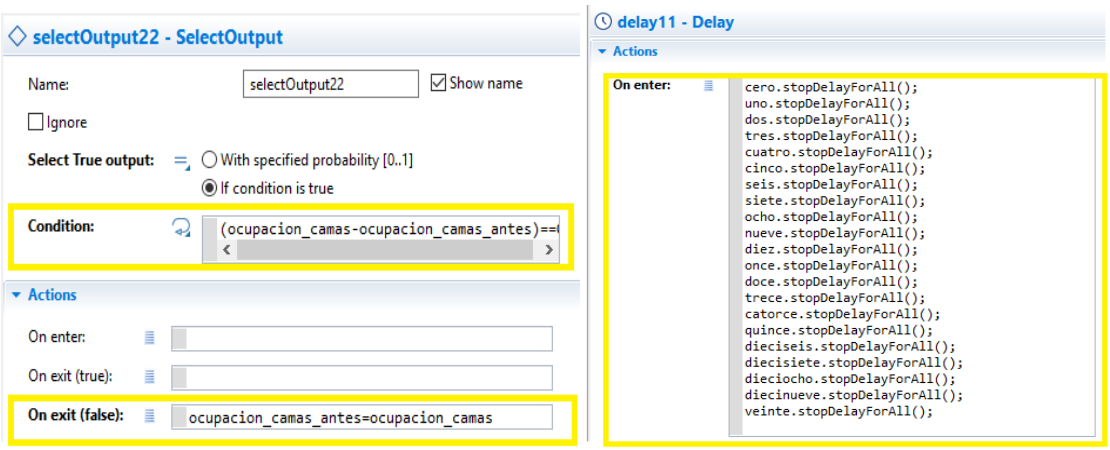


Figura 82: *Elementos a modificar en los bloques Delay y SelectOutput del bucle del cálculo de la frecuencia de ocupación de camas.*

De este modo, el tiempo que permanezcan las entidades en cada bloque *delay* que representa la ocupación de las camas es almacenado en su correspondiente variable, lo que va a permitir calcular cual es la frecuencia de ocupación de las camas de la UCI.

6. RESULTADOS

6.1. Escenarios a simular

Utilizamos el modelo de simulación para estudiar distintos escenarios. En primer lugar se va a simular el modelo sin altas prematuras. En este escenario todos los pacientes que no son exitus son dados de alta cuando están completamente estabilizados, es decir, cuando alcanzan el estado absorbente. Por otra parte, se va a simular el modelo con altas prematuras cambiando los coeficientes β de la distribución de probabilidad.

$$prob\_alta = \beta_0 + \sum_{i=1}^{3} \sum_{n=18}^{20} \beta_{in} \cdot Z_{in} =$$

$$= \beta_0 + \beta_1 X_1 1_{\{y=20\}} + \beta_2 X_1 1_{\{y=19\}} + \beta_3 X_1 1_{\{y=18\}} + \beta_4 X_2 1_{\{y=20\}} + \beta_5 X_2 1_{\{y=19\}}$$
$$+ \beta_6 X_2 1_{\{y=18\}} + \beta_7 X_3 1_{\{y=20\}} + \beta_8 X_4 1_{\{y=19\}} + \beta_9 X_3 1_{\{y=18\}}$$

Escenario 1. Sin altas prematuras.

En este primer escenario se va a simular el modelo sin tener en cuenta las altas prematuras, es decir, suponiendo que todos los pacientes son dados de alta cuando están completamente estabilizados.

El objetivo de simular este escenario es observar cómo se comporta el sistema con el objetivo de poder compararlo posteriormente con los distintos escenarios que incorporan la salida de pacientes de manera prematura de la UCI.

Escenario 2. Coeficientes β para un porcentaje de rechazo de pacientes del 4.5%.

El segundo escenario que se va a simular contempla los coeficientes β del artículo [3] para un porcentaje de rechazos del 4,5%. Estos coeficientes deben introducirse en la fórmula correspondiente al cálculo de la probabilidad de alta prematura.

β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9
-10	8.65	6.45	3.8	6.45	6.45	2.65	6.45	0.5	0.5

Escenario 3. Coeficientes β para un porcentaje de rechazo de pacientes del 3%.

Por otra parte, el tercer escenario que se va a simular contiene los coeficientes β para un porcentaje de rechazo de pacientes del 3% correspondientes al artículo [3]. A priori, este escenario representa una política de altas más agresiva y por lo tanto un porcentaje de rechazo de pacientes menor.

β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9
-10	18.4	18.4	18.4	18.4	18.4	15.55	15.55	12.75	4.1

Escenario 4. Modificación de coeficientes β

Por último, el tercer escenario que se va a simular contiene unos coeficientes β propuestos, que no aparecen en el artículo pero que mantienen la misma proporcionalidad que los que aparecen en él. El objetivo es ver como varían los resultados de este escenario con respecto a los anteriores según los coeficientes que se escogen.

Cabe destacar que existe una relación de monotonía entre los valores de los coeficientes que se debe tener en cuenta a la hora de darles valor. Esto ocurre debido a que estos coeficientes van multiplicados a los valores del vector de llegadas programadas y el peso de algunos de ellos debe ser mayor que el de otros como por ejemplo el coeficiente β_3 no puede ser mayor que β_2 y β_4 no puede ser mayor que β_3.

β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9
-20	6.90	6.90	5.40	5.20	4.1	3.0	1.4	0.8	0.2

6.2. Porcentaje de pacientes de grupo 1 y grupo 2 rechazados.

En este apartado, se muestran los resultados correspondientes al porcentaje de pacientes rechazados para cada tipo de paciente y en total.

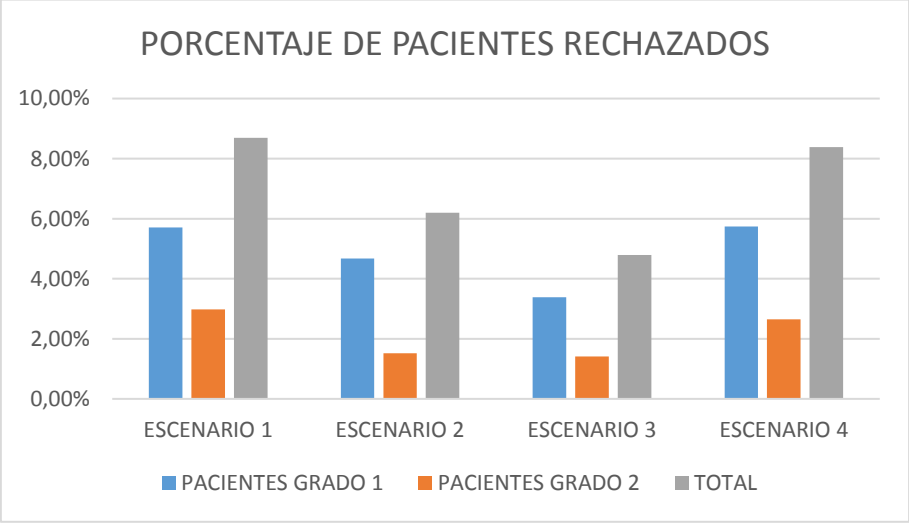
Analizando los resultados obtenidos se puede observar una disminución del pocentaje de pacientes rechazados de los escenarios que incluyen altas prematuras en el modelo (escenarios 2,3 y 4) respecto al que no se producían altas prematuras (escenario 1). Esto indica un aparente buen comportamiento del modelo de simulación.

Por otra parte, comparando los escenarios 2 y 3 se puede observar como el escenario 2 tiene un porcentaje de rechazo de pacientes mayor que el escenario 3 y esto se debe a que los coeficientes que se han utilizado. En el escenario 2 se han utilizado coeficientes β correspondientes a una política de altas más conservadora que en el escenario 3 y por ello, el porcentaje de pacientes rechazados es mayor.

En general, se puede detectar una relación razonable entre los resultados obtenidos y los escenarios que se han planteado.

PORCENTAJE DE PACIENTES RECHAZADOS	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4
PACIENTES GRUPO 1	5,71%	4,68%	3,38%	5,74%
PACIENTES GRUPO 2	2,98%	1,52%	1,41%	2,65%
TOTAL	8,69%	6,20%	4,79%	8,39%

Tabla 4: *Porcentaje de pacientes rechazados para cada tipo de paciente y por cada escenario.*



Gráfica 1: *Representación del porcentaje de pacientes rechazados.*

6.3. Tiempo de estancia de los pacientes de grupo 1 y grupo 2.

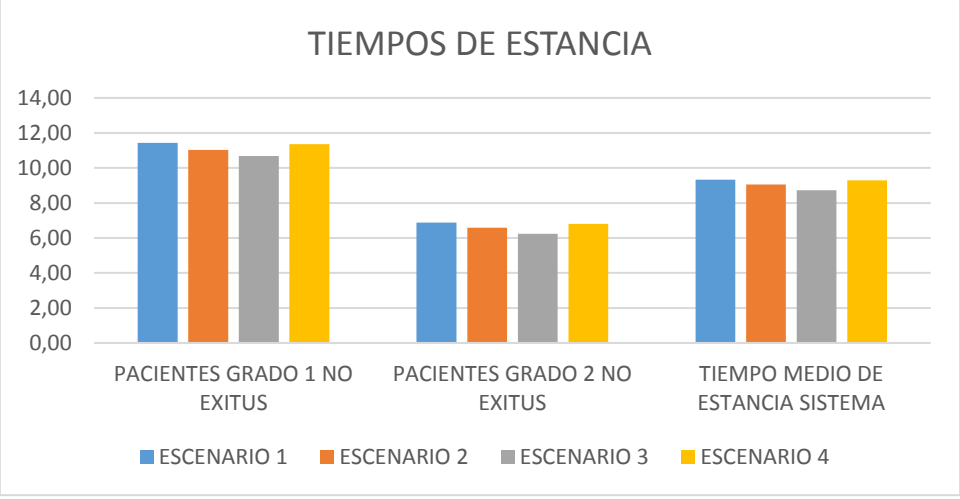
En este apartado, se muestran los resultados correspondientes a los tiempos de estancia en días para cada tipo de paciente y por cada escenario.

Analizando los resultados obtenidos, se puede observar que la variación de cada uno de los tiempos de un escenario a otro es pequeña.

Esto se debe principalmente a que en los escenarios se está modificando la manera de realizar el cálculo de las altas prematuras y no otros parámetros tales como la tasa de llegadas o el tiempo de estancia en las distintas fases de los pacientes. En realidad, la única variación que va a aparecer en los tiempos de estancia está directamente relacionada con el tiempo que se acorta en la fase 5 de estancia de los pacientes.

TIEMPOS DE ESTANCIA (días)	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4
PACIENTES GRADO 1 NO EXITUS	11,43	11,02	10,67	11,36
PACIENTES GRADO 2 NO EXITUS	6,88	6,59	6,23	6,81
TIEMPO MEDIO DE ESTANCIA SISTEMA	9,34	9,05	8,73	9,29

Tabla 5: *Tiempos de estancia para cada tipo de paciente y por cada escenario.*



Gráfica 2: *Representación de los tiempos de estancia.*

6.4. Porcentaje de pacientes dados de alta prematuramente total, de grupo 1 y de grupo 2

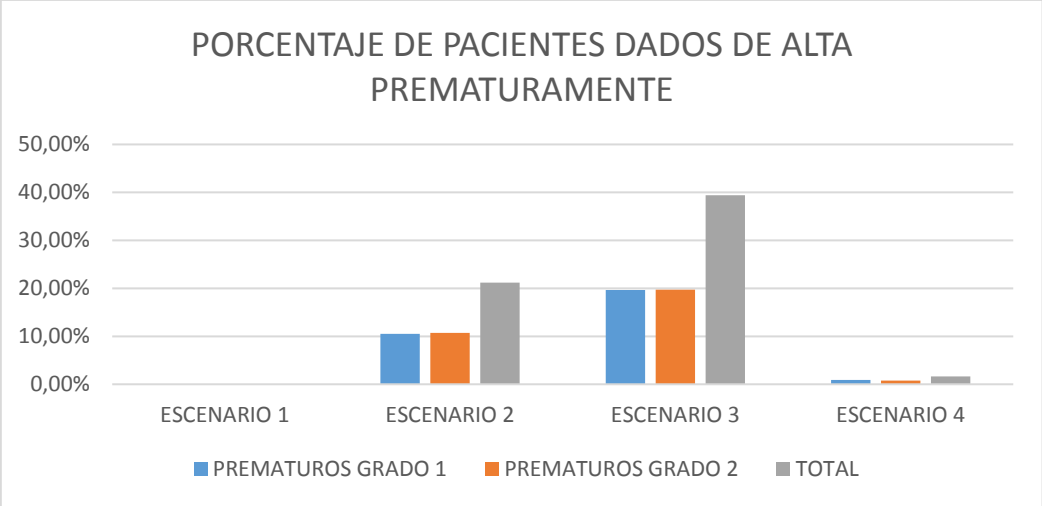
Analizando los resultados obtenidos, se puede observar una variación elevada del número de pacientes dados de alta prematuramente en el sistema.

Al igual que ocurre con los resultados anteriores, las variaciones son acordes a los escenarios planteados, ya que, con una política de altas más agresiva como la del escenario 3 el número de pacientes cuya estancia se acorta debe ser mayor. Por el contrario, cuando la política de altas es más conservadora el número de pacientes prematuros disminuyen notablemente.

Por último, como cabe esperar en el escenario uno no hay pacientes con estancia acortada o prematuros, ya que el escenario contempla una UCI sin altas prematuras.

PORCENTAJE DE PACIENTES DADOS DE ALTA PREMATURAMENTE	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4
DE GRUPO 1	0,00%	10,50%	19,67%	0,88%
DE GRUPO 2	0,00%	10,72%	19,72%	0,78%
TOTAL	0,00%	21,22%	39,40%	1,66%

Tabla 6: *Porcentaje de pacientes dados de alta prematuramente para cada tipo de paciente y por cada escenario.*



Gráfica 3: *Representación del porcentaje de pacientes dados de alta prematuramente.*

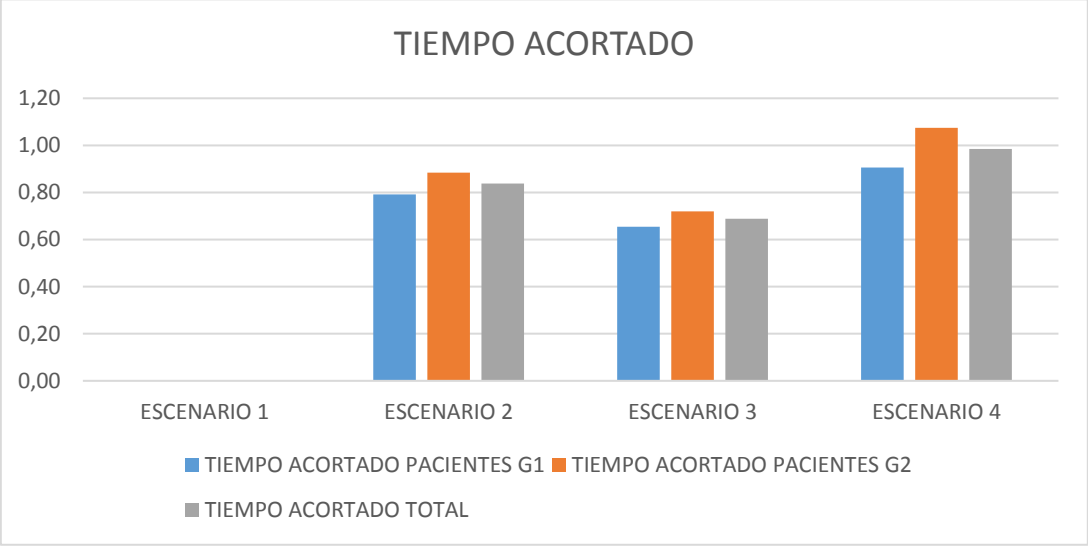
6.5. Tiempo medio acortado: total y para los pacientes de grupo 1 y grupo 2

Estos resultados están directamente relacionados con los resultados de los dos apartados anteriores, es decir, los del tiempo medio en el sistema y el porcentaje de pacientes prematuros.

Los resultados por lo tanto son perfectamente acordes a los escenarios planteados. En el escenario uno no existe tiempo acortado debido a que el escenario no contempla altas prematuras. Por otra parte, en el resto de escenarios se puede observar un mayor tiempo acortado según más agresiva sea la política de altas.

TIEMPOS ACORTADO (días)	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4
TIEMPO ACORTADO PACIENTES G1	0,00	0,79	0,66	0,91
TIEMPO ACORTADO PACIENTES G2	0,00	0,88	0,72	1,07
TIEMPO ACORTADO TOTAL	0,00	0,84	0,69	0,99

Tabla 7: *Tiempos de estancia acortados para cada tipo de paciente y por cada escenario.*



Gráfica 4: *Representación de los tiempos de estancia acortados.*

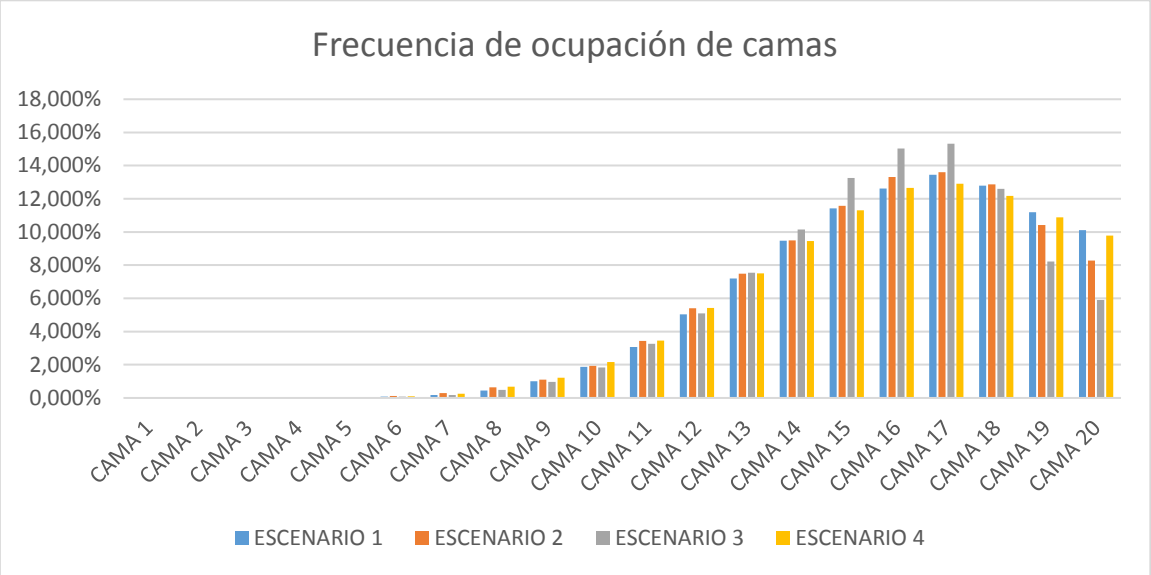
6.6. Frecuencia de ocupación de camas

Los resultados de la frecuencia de la ocupación de camas muestran una mayor frecuencia de ocupación de 16, 17 y 18. Cabe destacar que la mayor frecuencia de ocupación es de 19 y 20; esto está directamente ligado a la manera en la que está programado el modelo.

Esto se debe principalmente a que el modelo está programado con distribuciones de ocupación de camas obtenidas con datos reales dónde los médicos dan altas prematuras para evitar la saturación en la UCI. De este modo, el número de ocupación más frecuente de camas es de 16, 17 y 18.

FRECUENCIA DE OCUPACION DE CAMAS	ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4
CAMA 1	0,001%	0,010%	0,013%	0,004%
CAMA 2	0,009%	0,008%	0,003%	0,003%
CAMA 3	0,020%	0,001%	0,007%	0,015%
CAMA 4	0,009%	0,009%	0,008%	0,001%
CAMA 5	0,025%	0,039%	0,017%	0,025%
CAMA 6	0,073%	0,116%	0,082%	0,093%
CAMA 7	0,178%	0,287%	0,168%	0,247%
CAMA 8	0,445%	0,633%	0,483%	0,673%
CAMA 9	1,009%	1,107%	0,967%	1,222%
CAMA 10	1,873%	1,926%	1,842%	2,160%
CAMA 11	3,068%	3,434%	3,269%	3,451%
CAMA 12	5,045%	5,410%	5,100%	5,428%
CAMA 13	7,207%	7,483%	7,555%	7,513%
CAMA 14	9,467%	9,498%	10,152%	9,464%
CAMA 15	11,428%	11,585%	13,256%	11,303%
CAMA 16	12,611%	13,305%	15,038%	12,658%
CAMA 17	13,448%	13,598%	15,320%	12,904%
CAMA 18	12,787%	12,864%	12,598%	12,167%
CAMA 19	11,192%	10,412%	8,218%	10,887%
CAMA 20	10,105%	8,274%	5,902%	9,782%

Tabla 8: *Frecuencia de ocupación de las camas.*



Gráfica 5: *Representación de la frecuencia de ocupación de las camas.*

7. CONCLUSIONES

Como se ha comentado al inicio de la memoria, el objetivo principal de este Trabajo Fin de Master es construir un modelo de simulación basada en agentes que represente el funcionamiento de las UCIs y que pueda utilizarse para el estudio de la toma de decisiones médicas en relación con el alta de pacientes. El modelo que se ha construido está basado en la literatura actual que hay acerca del tema, más concretamente, se ha utilizado la información que aparece en [3].

En cuanto al proceso que se ha llevado a cabo para realizar el modelo de simulación se puede decir que se han completado todas las etapas que se han mencionado al inicio de la memoria (apartado 2.1.). Cabe destacar que alguna de estas etapas ha requerido mayor dedicación que otras. Las etapas del planteamiento del problema y la recogida de datos han sido relativamente cortas ya que desde un principio ambas estaban perfectamente definidas. Se puede decir que la etapa que más tiempo ha requerido ha sido la de programación. Esto se debe principalmente al desconocimiento de la utilización del software de simulación Anylogic. Por lo tanto, el aprendizaje obtenido en lo que a programación en el software se refiere ha sido elevado.

La realización de experimentos de simulación mediante un ejemplo ilustrativo ha servido para analizar el funcionamiento del modelo. Los datos que se han empleado a modo de ejemplo ilustrativo así como las medidas de funcionamiento que se han obtenido del mismo han sido extraídos de [3].

Por último, se puede decir que este modelo realizado puede ser utilizado para cualquier UCI si se modifican las variables y los parámetros necesarios. Además, la manera en la que se han representado la toma de decisiones de los médicos muestra una línea de investigación abierta de ya que pueden incorporarse nuevos elementos a este proceso de toma de decisiones. En el trabajo *"Implementación de simulación basada en agentes en modelo de simulación de una unidad de cuidados intensivos"* se añaden nuevos elementos al proceso de toma de decisiones de los médicos tomando como base el modelo realizado en el presente trabajo.

8. BIBLIOGRAFÍA

[1] Azcárate C.; Mallor F. "***DOCUMENTACIÓN DE LA ASIGNATURA SIMULACIÓN PARA LA TOMA DE DECISIONES"***, *2019.*

[2] Sitio web. **"*COMPONENTES Y ETAPAS DE LA SIMULACIÓN*".** Retrieved 30 July 2019, from https://aguinagasimulacion.wordpress.com/2016/09/06/componentes-y-etapas-de-la-simulacion/

[3] Azcárate C.; Esparza L.; Mallor F. ***"THE PROBLEM OF THE LAST BED: CONTEXTUALIZATION AND A NEW SIMULATION FRAMEWORK FOR ANALYZING PHYSICIAN DECISIONS"***. *Omega*, 2019, pp. 1-20.

[4] García - Valdecasas, J.. "***LA SIMULACIÓN BASADA EN AGENTES: UNA NUEVA FORMA DE EXPLORAR LOS FENÓMENOS SOCIALES".*** *Revista española de investigaciones sociológicas.* Reis 136, OctubreDdiciembre 2011, pp. 91-110

[5] Cabrera, E.; Taboadab, M.; Iglesias, Mª L.; Epelde, F.; Luque, Emilio. "***OPTIMIZATION OF HEALTHCARE EMERGENCY DEPARTMENTS BY AGENT-BASED SIMULATION".*** *International Conference on Computational Science, (ICCS 2012),* Procedia Computer Science 9 (2012) pp.1464 – 1473.

[6] Viana, J.; Margrethe Ziener, V.; Sommer Holhjem, M. ***"OPTIMIZING HOME HOSPITAL HEALTH SERVICE DELIVERY IN NORWAY USING A COMBINED GEOGRAPHICAL INFORMATION SYSTEM, AGENT BASED, DISCRETE EVENT SIMULATION MODEL".*** *Proceedings of the Winter Simulation Conference,* December 2017, Article No.: 128 pp. 1–12.

[7] Al Fatah, J.; Alshaban, A.; Holmgren, J.; Petersson, J. "***AN AGENT-BASED SIMULATION MODEL FOR ASSESSMENT OF PREHOSPITAL TRIAGE POLICIES CONCERNING DESTINATION OF STROKE PATIENTS".*** *The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2018),* Article No.: 141 (2018) pp. 405-412.

[8] Fu, M.; Glover, F.; April, J. **"*SIMULATION OPTIMIZATION: A REVIEW, NEW DEVELOPMENTS, AND APPLICATIONS*".** *Proceedings of the Winter Simulation Conference,* 2005, pp. 83-95.

[9] Sitio web. "***HELP - ANYLOGIC SIMULATION SOFTWARE (2019)".*** Retrieved 07 September 2019, from https://help.anylogic.com/index.jsp

[10] Sitio web. "***JAVA IF ... ELSE (2019)".*** Retrieved 17 October 2019, from https://www.w3schools.com/java/java_conditions.asp

[11] Sitio web. "***NEWEST 'ANYLOGIC' QUESTIONS".*** Retrieved 16 May 2019, from https://stackoverflow.com/questions/tagged/anylogic